



A/UX[®] Local System Administration

Production draft
1/1/89

Linda Kinnier
Developer Technical Publications

This Document contains preliminary information.

It does not include:

- table of contents
- final technical information
- final editorial corrections
- final art
- a glossary

© Apple Computer, Inc. 1988

 APPLE COMPUTER, INC.

This manual is copyrighted by Apple or by Apple's suppliers, with all rights reserved. Under the copyright laws, this manual may not be copied, in whole or in part, without the written consent of Apple Computer, Inc. This exception does not allow copies to be made for others, whether or not sold, but all of the material purchased may be sold, given, or lent to another person. Under the law, copying includes translating into another language.

The Apple logo is a registered trademark of Apple Computer, Inc. Use of the "keyboard" logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

© Apple Computer, Inc., 1988
20525 Mariani Avenue
Cupertino, CA 95014
(408) 996-1010

Apple, the Apple logo, LaserWriter, and Macintosh are registered trademarks of Apple Computer, Inc.

ITC Avant Garde Gothic, ITC Garamond, and ITC Zapf Dingbats are registered trademarks of International Typeface Corporation.

Microsoft is a registered trademark of Microsoft Corporation.

POSTSCRIPT is a registered trademark of Adobe Systems Incorporated.

Varityper is a registered trademark, and VT600 is a trademark, of AM International, Inc.

Simultaneously published in the United States and Canada.

9/12/88

LIMITED WARRANTY ON MEDIA AND REPLACEMENT

If you discover physical defects in the manuals distributed with an Apple product or in the media on which a software product is distributed, Apple will replace the media or manuals at no charge to you, provided you return the item to be replaced with proof of purchase to Apple or an authorized Apple dealer during the 90-day period after you purchased the software. In addition, Apple will replace damaged software media and manuals for as long as the software product is included in Apple's Media Exchange Program. While not an upgrade or update method, this program offers additional protection for up to two years or more from the date of your original purchase. See your authorized Apple dealer for program coverage and details. In some countries the replacement period may be different; check with your authorized Apple dealer.

ALL IMPLIED WARRANTIES ON THE MEDIA AND MANUALS, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF THE ORIGINAL RETAIL PURCHASE OF THIS PRODUCT.

Even though Apple has tested the software and reviewed the documentation, **APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO SOFTWARE, ITS QUALITY, PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS SOFTWARE IS SOLD "AS IS," AND YOU THE PURCHASER ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND PERFORMANCE.**

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT IN THE SOFTWARE OR ITS DOCUMENTATION, even if advised of the possibility of such damages. In particular, Apple shall have no liability for any programs or data stored in or used with Apple products, including the costs of recovering such programs or data.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

9/12/88

A/UX Local System Administration

Contents

Preface	Conventions Used in This Manual
Chapter 1	Managing the A/UX System—An Introduction
Chapter 2	System Startup and Shutdown
Chapter 3	User and Group Administration
Chapter 4	Backing Up Your System
Chapter 5	Preparing an Apple HD SC for A/UX
Chapter 6	Managing Disks
Chapter 7	Managing Other Peripheral Devices
Chapter 8	Checking the A/UX File System: <code>fsck</code>
Chapter 9	System Accounting Package
Chapter 10	System Activity Package
Appendix A	The UUCP System
Notes	
Index	

BLANK PAGE

Preface

Conventions Used In This Manual

A/UX® manuals follow certain conventions regarding presentation of information. Words or terms that require special emphasis appear in specific fonts within the text of the manual. The following sections explain the conventions used in this manual.

Significant fonts

Words that you see on the screen or that you must type exactly as shown appear in `Courier` font. For example, when you begin an A/UX work session, you see the following on the screen:

```
login:
```

The text shows `login:` in `Courier` typeface to indicate that it appears on the screen. If the next step in the manual is

```
Enter start
```

`start` appears in `Courier` to indicate that you must type in the word. Words that you must replace with a value appropriate to a particular set of circumstances appear in *italics*. Using the example just described, if the next step in the manual is

```
login: username
```

you type in your name—*Laura*, for example— so the screen shows:

```
login: Laura
```

New terminology appears in boldface. Common A/UX terms are defined in the glossary of *Getting Started with A/UX*.

Key presses

Certain keys are identified with names on the keyboard. These modifier and character keys perform functions, often in combination with other keys. In the manuals, the names of these keys appear in the format of an Initial Capital letter followed by SMALL CAPITAL letters.

The list that follows provides the most common keynames.

RETURN	DELETE	SHIFT	ESCAPE
OPTION	CAPS LOCK	CONTROL	

For example, if you enter

Applee

instead of

Apple

you would position the cursor to the right of the word and press the DELETE key once to erase the additional *e*.

For cases in which you use two or more keys together to perform a specific function, the keynames are shown connected with hyphens. For example, if you see

Press CONTROL-C

you must press CONTROL and C simultaneously (CONTROL-C normally cancels the execution of the current command).

Terminology

In A/UX manuals, a certain term can represent a specific set of actions. For example, the word *Enter* indicates that you type in an entry and press the RETURN key. If you were to see

Enter the following command: whoami

you would type whoami and press the RETURN key. The system would then respond by identifying your login name.

Here is a list of common terms and their corresponding actions.

expand tab (:); lw(1i)fB lw(3i)fB lfB l . _ Term:Action _
T{ Enter T}:T{ Type in the entry and press the RETURN key T}

Press:T{ Press a *single* letter or key *without* pressing the RETURN key T}

Type:T{ Type in the letter or letters *without* pressing the RETURN key T}

expand tab (:); lw(1i)fB lw(3i)fB lfB l . _ Term:Action _
Click:T{ Press and then immediately release the mouse button T}

Select:T{ Position the pointer on an item and click the mouse button T}

Drag:T{ Position the pointer on an icon, press and hold down the mouse button while moving the mouse. Release the mouse button when you reach the desired position. T}

Choose:T{ Activate a command title in the menu bar. While holding down the mouse button, drag the pointer to a command name in the menu and then release the mouse button. An example is to drag the File menu down until the command name Open appears highlighted and then release the mouse button. T}

Syntax notation

A/UX commands follow a specific order of entry. A typical A/UX command has this form:

command [*flag-option*] [*argument*] . . .

The elements of a command have the following meanings.

expand tab (:); lw(1i)fB lw(3i)fB l l . _ Element:Description _
command:Is the command name.

flag-option:T{ Is one or more optional arguments that modify the command. Most flag-options have the form
[-opt...]

where opt is a letter representing an option. Commands can take one or more options. T}

argument:T{ Is a modification or specification of the command; usually a filename or symbols representing one or more filenames. T}

brackets ([]):T{ Surround an optional item—that is, an item that you do not need to include for the command to execute. T}

ellipses (...):T{ Follow an argument that may be repeated any number of times. T}

expand tab (:); lw(1i)fB lw(3i)fB 11. _ Element:Description _

For example, the command to list the contents of a directory (`ls`) is followed below by its possible flag options and the optional argument *names*.

```
ls [-R] [-a] [-d] [-C] [-x] [-m] [-l] [-L]
    [-n] [-o] [-g] [-r] [-t] [-u] [-c] [-p] [-F]
    [-b] [-q] [-i] [-s] [names]
```

You can enter

```
ls -a /users
```

to list all entries of the directory `/users`, where

<code>ls</code>	Represents the command name
<code>-a</code>	Indicates that <i>all</i> entries of the directory be listed
<code>/users</code>	Names which directory is to be listed

Command reference notation

The manuals *A/UX Command Reference*, *A/UX Programmer's Reference*, and *A/UX System Administrator's Reference* contain references for commands, programs, and other related information. Reference material is organized within these manuals by section numbers. The standard A/UX cross-reference notation is

cmd(sect)

where *cmd* is the name of the command, file, or other facility; *sect* is the section number where the entry resides.

- ☐ Commands followed by section numbers (1M), (7), or (8) are listed in *A/UX System Administrator's Reference*.
- ☐ Commands followed by section numbers (1), (1C), (1G), (1N), and (6) are listed in *A/UX Command Reference*.
- ☐ Commands followed by section numbers (2), (3), (4), and (5) are listed in *A/UX Programmer's Reference*.

For example,

```
cat(1)
```

refers to the command `cat`, which is described in Section 1 of *A/UX Command Reference*.

References can be also called up on the screen. The `man` or `apropos` command displays pages from reference manuals directly on the screen. For example, enter the command

```
man cat
```

In this example, the manual page for the `cat` command, including its description, syntax, options, examples and other pertinent information, appears on the screen. To exit, continue pressing the space bar until you see your command prompt or press `Q` at any time to return immediately to your command prompt.

An A/UX manual often refers to information discussed in another manual in the suite. The format for this type of reference is "Chapter Title," *Name of Manual*.

PLACE TAB
HERE

INTRODUCTION

BACK OF TAB

Chapter 1

Managing the A/UX System

An Introduction

Contents

1. The system administrator	1
2. A/UX system administration manuals	1
3. For the new system administrator	2
4. Administrative logins on the A/UX system	4
4.1 Administrative groups	6
5. The complete contents of A/UX	6

BLANK PAGE

Chapter 1

Managing the A/UX System

An Introduction

1. The system administrator

This book is for the A/UX® system administrator. The system administrator is responsible for making sure that the system runs smoothly. The administrator typically performs the following functions:

- starting up and shutting down the system
- adding, modifying, and removing user accounts
- backing up the system
- adding and managing peripheral devices
- locating and resolving inconsistencies in the file system
- monitoring system activity
- monitoring user activity
- setting up and maintaining communication channels with other computer systems

With the exception of communicating with other systems, all of these procedures affect a single computer and are considered **local system administration**. These procedures are described in this manual. For information on setting up and administering networks, see *A/UX Network System Administration*.

2. A/UX system administration manuals

The information you need for managing single (not networked) A/UX systems is presented in two manuals:

- *A/UX Local System Administration* (which you are currently reading) contains ten chapters. This chapter is an introduction and overview, and the remaining nine chapters are devoted to the topics in the preceding bulleted list. These chapters include

step-by-step instructions that guide you through routine procedures as well as background information about the A/UX environment. The background information is useful for analyzing and handling unexpected situations or error conditions. First-time administrators should read these chapters in sequence.

- *A/UX System Administrator's Reference* contains details and variations on the standard commands and procedures. It summarizes the main system administration commands but does not contain introductory or overview material. It is recommended that you read *A/UX Local System Administration* first and use *A/UX System Administrator's Reference* for quick reference once you understand how a command works. The commands documented in *A/UX System Administrator's Reference* are cited as *command-name(1M)*. You will also find references to commands cited as *command-name(1)*. These are contained in *A/UX Command Reference*.

3. For the new system administrator

This book combines theory and practice to help you understand what to do as well as why. If you are not an experienced system administrator, note the following suggestions before you begin:

- Log in as the root user (*root*, *rootcsh*, or *rootksh*) only when absolutely necessary. The root user has special privileges, and you can perform functions as the root user that you can't perform if you are logged in as a normal user. Thus, as a normal user you have some protection against making mistakes that might affect the operation of your system. If you make mistakes as the root user, you can crash (or damage) your system.
- While you're logged in as the root user, record everything you do in a system administrator's log. This log should contain an exact description of what you do, why you do it, when you do it, and what effect it had.

- Make backups of all important system files before you change them. For example, if you are about to modify the `/etc/inittab` file, first enter the command

```
cp /etc/inittab /etc/inittab.old
```

Then, if your modifications prove unworkable, it is an easy matter to return the system to its prior state:

```
mv /etc/inittab.old /etc/inittab
```

- Use `cron` to automate routine system administration duties or to remind yourself to do these duties. See Chapter 6, "Managing Disks," Chapter 9, "System Accounting Package," and Chapter 10, "System Activity Package." Also see `cron(1M)` in *A/UX System Administrator's Reference* and `crontab(1)` in *A/UX Command Reference*.
- Use `find` to locate large and dormant files and directories. Large, unused files are often not needed and consume valuable storage space. When you find such files, you can either remove them (for example, if they are core images of crashed programs), truncate them to zero length (if they are log files), transfer them to tape or diskette archive (if you are even remotely likely to want to use them again in the future), or compress them to a smaller size. See Chapter 4, "Backing Up Your System," or refer to `find(1)` in *A/UX Command Reference*.
- Watch for files that grow (in general, any file containing the letters `log` or `LOG` as part of its name). The system appends information to these files, and they can grow to excessive size. You should regularly delete or truncate these files after backing them up. See Chapter 6, "Managing Disks."
- Make frequent backups of your files. Backups are your insurance if something should go wrong. How often you perform backups depends on how much activity there is on your system and how much work you're willing to lose. See Chapter 4, "Backing Up Your System."
- If you are administering a multi-user system, choose low-activity times to perform potentially disruptive tasks. Early morning and late evening are good choices. Check to see who is logged in

before you begin; if anyone is actively running processes, notify them with the `wall` command. See `wall(1M)` in *A/UX System Administrator's Reference*.

- If a catastrophe occurs:

1. Think carefully about what happened.
2. Plan your next action before actually doing it.
3. Anticipate the consequences of implementing your plan.
4. Act.
5. Write down in your system log what happened, what you did, and how the system responded.

One of the worst things you can do in the event of a system catastrophe (such as a loss of important data) is to act without first considering the consequences of your actions.

4. Administrative logins on the A/UX system

The standard distribution of the A/UX system contains one normal user login account, `start`, whose home directory is `/users/start`. This login account is provided to give new users a place to store files used in the introductory tutorials; see *Getting Started with A/UX*.

All other logins on the standard distribution are administrative logins, login accounts that are used by system administrators or by A/UX programs to perform specialized system tasks. The administrative logins are as follows:

`root`
`rootcsh`
`rootksh`

Logins for the root user. These logins have identical permissions in the A/UX system; they differ only with regard to the startup program, which is, respectively, the Bourne shell, the C shell, and the Korn shell. The home directory for all three logins is the root directory, `/`.

`daemon`

The owner of certain noninteractive, background processes that handle persistent system services, such as network communication.

- bin** The owner of most normal A/UX commands and the system directories in which those commands are stored.
- sys** The owner of certain A/UX system files, such as `/etc/zoneinfo` and the files used by `autorecovery`.
- adm** The owner of most system accounting programs and directories, in particular `/usr/adm`. See Chapter 9, "System Accounting Package."
- nuucp**
The login assigned to incoming UUCP requests is `nuucp`. See Appendix A, "The UUCP System," for further information.
- uucp**
The owner of programs and directories associated with the UUCP communications package.
- lp** The owner of commands and processes associated with the `lp` spooling and printing package. See "Setting Up a Printer" in Chapter 7, "Managing Other Peripheral Devices," for more information on the `lp` system and the `lp` login.
- ftp** The owner of files and directories associated with `ftp`, a file transfer program. See *A/UX Communications User's Guide* for a description of the `ftp` command.
- nobody**
A login assigned as the default login for remote root access under NFS. See *A/UX Network System Administration* for further information.
- who** A login that allows users who do not have accounts on a system to see who is currently logged into the system. There is no password for this login. The startup program is simply `/bin/who`.

4.1 Administrative groups

Like administrative logins, administrative groups help you perform specialized system tasks and are provided in the standard distribution of A/UX. An administrative group lets you avoid running a program as the root user by instead running it with the group ID of the group provided for this special purpose. This reduces the security risk inherent in running programs as the root user.

The administrative groups provided in A/UX are `sys`, `bin`, and `root`.

5. The complete contents of A/UX

A/UX consists of an enormous number of files. If you wonder exactly what those files are, you can find the answer in the complete list of distributed A/UX files. This list is provided in a file called `/FILES`, which gives the full pathname of every A/UX system file along with a short description of its use. The list is a valuable resource when you want to quickly look up the purpose of a particular system file.

PLACE TAB
HERE

STARTUP AND
SHUTDOWN

BACK OF TAB

Chapter 2

System Startup and Shutdown

Contents

1. Introduction	1
1.1 What is the Stand-Alone Shell?	1
1.2 Invoking sash	2
1.3 What else can sash do?	2
2. Starting up the A/UX system	3
2.1 The default booting sequence	3
2.2 When the system automatically reboots	5
2.2.1 When fsck automatically reboots the system	5
2.2.2 When autoconfig automatically reboots the system	6
3. Changing the startup device and application	7
3.1 Steps to change the startup device	7
3.2 Technical details	10
3.2.1 The natural order of startup devices	10
3.2.2 How A/UX boots from a hard disk	10
3.3 Changing the startup application	11
4. Setting system-specific details	12
4.1 Setting the system time	12
4.1.1 Time in A/UX and the Macintosh OS	12
4.1.2 Resetting after a hardware problem.	17
4.1.3 Resetting after moving a system to a different time zone	17
4.1.4 Resetting for Daylight Savings Time	19
4.1.5 Overriding the default time zone	20
4.2 Changing the message of the day	20
4.3 Renaming the system	21
4.4 About single-user mode	21

4.5	Entering multi-user mode	21
4.5.1	Entering multi-user mode manually	22
4.5.2	Entering multi-user mode automatically at boot time	23
4.6	Changing kernel parameters	24
4.6.1	Systems with 4 or more megabytes of main memory	24
4.7	Initial processes: <code>/etc/inittab</code>	25
4.7.1	<code>/etc/inittab</code> entry format	26
4.8	Changing run levels: <code>init</code>	28
5.	Shutting down the A/UX system	29
5.1	Using the <code>shutdown</code> program	30
5.2	Shutting down the A/UX system manually	32
5.3	If your system hangs	34

Figures

Figure 2-1.	The Control Panel	8
Figure 2-2.	General dialog box	9
Figure 2-3.	GMT biases	15

Chapter 2

System Startup and Shutdown

1. Introduction

This chapter describes how to start up and shut down the A/UX operating system. These procedures are largely automatic, but they do require supervision and possible intervention by the system administrator. These procedures may also be customized to suit local needs.

This chapter also briefly describes the Stand-Alone Shell (`sash`). The Stand-Alone Shell is a Macintosh® program that is invoked to boot A/UX. Like other shells, `sash` is a command interpreter with a command language and environment similar to (but simpler than) that of the Bourne shell. You will use `sash` primarily to startup the A/UX system. This is described in "The Default Booting Sequence" in this chapter. Additionally, you use `sash` to run the autorecovery facility, and key stand-alone versions of A/UX utilities when you troubleshoot A/UX. See "The Stand-Alone Shell" and `sash(8)` in *A/UX System Administrator's Reference*.

1.1 What Is the Stand-Alone Shell?

The Stand-Alone Shell (`sash`) is merely a Macintosh Operating System (OS) program. The Macintosh computers always come up under the Macintosh OS. It is `sash`'s job to hand over control of the machine from the Macintosh OS to A/UX. If you don't suspect that you'll want to be able to boot A/UX from a disk, then there is no need for the disk to contain `sash`.

The small distinction in names might be confusing, but `sash` (the Macintosh program) resides in a small Macintosh partition (called SASH) on a disk containing A/UX. To remember the differences, keep in mind that `sash` is a part of SASH.

A partition is a portion of a disk set aside for a specific use. Its value is in separating subsets of information for easier management, much as a file system does. By definition, different operating systems must

occupy different partitions. For the Macintosh and A/UX operating systems to reside on the same disk, they must occupy separate partitions. Because `sash` is a Macintosh program to start up A/UX, `sash` resides in a small Macintosh partition (called SASH) on a disk that typically is otherwise taken up by A/UX.

1.2 Invoking `sash`

If you are running A/UX, invoke `sash` by rebooting the computer as described under "Shutting Down the A/UX System." If you are running the Macintosh OS, invoke `sash` by double-clicking the A/UX `sash` icon. When the automatic startup screen appears and begins counting down to zero, click Cancel or press the keystroke sequence `COMMAND-SHIFT-.` (Command-Shift-period). This places you in the Stand-Alone Shell window, and you see the shell prompt:

```
sash#
```

You can change the prompt to any other string you like by redefining the shell variable `PS1`. For example, if you want the `sash` prompt to be `hello:`, enter

```
PS1='hello: '
```

1.3 What else can `sash` do?

The bilateral makeup of `sash` is evident when it is open on the screen. The pull-down menus above the `sash` window offer Macintosh-style interaction with the machine, and simultaneously the command-line prompt sits in the `sash` window, allowing you to give a subset of A/UX commands.

The `sash` program is called a stand-alone shell because it is very similar to the A/UX shells, yet it is not A/UX. It is a command interpreter and has a command language with shell variables, comments, input and output redirection, and built-in commands. Programs run from `sash` can be read from either a Macintosh or an A/UX file system.

The following is a list of the commands you can run in `sash`. Remember, these are not the A/UX commands, but rather special, functionally identical, stand-alone versions of the A/UX commands, written for use in `sash`.

cat	chgrp	chmod
cp	date	dd
ed	esch	fsck
kconfig	launch	ls
mkdir	mkfs	mknod
od	pname	rm

These programs let you work on the A/UX file system without running it. These programs are especially useful to troubleshoot A/UX. For example, you might edit `/etc/inittab` without running A/UX if you suspect that that file is causing you problems at boot time. You can list the contents of the A/UX root directory, by entering

```
ls -CF /
```

Similarly, the command

```
ls -CF /users/start
```

lists the contents of the `/users/start` directory.

To boot A/UX from the Stand-Alone Shell, enter

```
boot
```

For more information on the Stand-Alone Shell, see `sash(8)` in *A/UX System Administrator's Reference*.

2. Starting up the A/UX system

This section assumes you have already followed the instructions for setting up your Macintosh workstation given in *A/UX Installation Guide*. Once the computer is set up, you are ready to start up (boot) the A/UX operating system.

Both the default booting sequence and the procedure to change the startup device if you have more than one hard disk are described in the following sections.

2.1 The default booting sequence

This section describes the normal booting sequence for an A/UX system. By default, the computer starts up from the hard disk with the highest SCSI ID number. If your system has more than one hard disk, the sequence can vary only slightly. That variance is explained in the later section, "The Natural Order of Startup Devices."

You turn on power to the computer by pressing the Power-On key on the keyboard. The location of the key varies among keyboard models, but on all the keyboards the Power-On key is marked with a triangle pointing left.

A few seconds after you turn the power on, the screen should light up, the **Welcome to Macintosh** message should appear briefly, and if you followed the instructions in *A/UX Installation Guide* to set **sash** as the startup application, the Stand-Alone Shell displays the automatic startup screen. This sequence signals that the machine started up from a hard disk and then the Macintosh Operating System on that disk started up the startup application, which normally is **sash** on an A/UX system. In turn, **sash** began to launch A/UX by presenting the A/UX automatic startup screen.

Unless you click **Cancel**, the automatic startup screen counts down to zero, then launches A/UX. To speed things up, press **RETURN** or click **Go once**, and A/UX is launched immediately. When launching starts, the Stand-Alone Shell causes A/UX to take control of the computer.

After five seconds or so, the screen blinks, A/UX copyright and Release ID information is displayed, and the following prompt appears:

```
Do you want to check the root file system? (y or n)
[default: n]
```

Answer yes by entering

y

This invokes the **fsck** (file system checking) program. This is always the safe thing to do when you start up A/UX. The **fsck** program checks the A/UX root file system in several phases. This takes a couple of minutes, and at each phase **fsck** displays a message on your screen. If **fsck** finds any inconsistencies in the file system, it either automatically fixes them or prompts for your input (see Chapter 8, "Checking the A/UX File System: **fsck**" for further information).

If **fsck** has made any corrections to the file system, A/UX automatically reboots. Then **fsck** asks again if you want to check the root file system. It is prudent to respond **y** to this prompt to check the root file system again.

Next, `autoconfig` is invoked. This program may cause a few messages to be displayed, which vary according to your hardware configuration.

Now you are in single-user mode. The screen shows

```
INIT: SINGLE USER MODE
```

and the message of the day appears, followed by this message:

```
TERM = (mac2)
```

Press RETURN after this prompt to set the terminal type to Macintosh II.

If you already set your host name during the installation process (see "Renaming the System," later in this chapter, for the steps you take to change the host name), that host name is used by the shell prompt. The shell prompt also displays the name of the user, which in this example is `root`. For example, if your system's name is `linda` you will see this prompt:

```
linda.root#
```

That completes the booting sequence. A/UX defaults to single-user mode because this is most useful when you are first setting up your system. The next sections, "Changing the Startup Device" and "Setting System-Specific Details" describe, among other things, changes you can make to the booting sequence. If you want bypass single-user mode and bring the system into multi-user mode automatically, see the following section, "Entering Multi-User Mode Automatically at Boot Time."

2.2 When the system automatically reboots

A/UX automatically reboots in two situations. It reboots when `fsck` checks the file system and finds the system needs rebooting to correct a file system problem. And it reboots when `autoconfig` finds an inconsistency between the cards in the expansion slots and the device drivers in the kernel and determines the system needs a new kernel to correct the problem. The following subsections describe these situations.

2.2.1 When `fsck` automatically reboots the system

The `fsck` program runs after the system boots and before the system enters single-user mode. The purpose of `fsck` is to find problems in

the file system and correct them before they spread to other parts of the disk. If `fsck` makes any corrections to the file system, `fsck` reboots the system.

2.2.2 When `autoconfig` automatically reboots the system

The system runs `autoconfig` before entering single-user mode. The `autoconfig` program checks the consistency between the hardware in the expansion slots and the information contained in the kernel regarding software slot device drivers. The `autoconfig` program ensures that the information between the two are consistent. For example, if you had a Tape Backup 40SC unit attached to your system, and removed it between this and your last session at the computer, `autoconfig` would detect the change, remove the tape device driver from the kernel, and reboot the system with the new kernel. On the other hand, if the information is consistent between the expansion slots and the slot device drivers, `autoconfig` exits without rebooting the kernel.

If the kernel contains a device driver and the matching expansion card is not present, `autoconfig` rebuilds the kernel by removing the device driver from the kernel and booting the new kernel. This protects you from performing input and output operations to a non-existent card. But it also means that if you add a device driver for an expansion card but then forget to add the actual card, `autoconfig` will remove the device driver when the system boots. When you later add the matching card, you will need to build a new kernel using the `launch newunix` command from `sash`.

If the kernel contains a slot device driver and the matching slot card is present in a different slot than the kernel expected, `autoconfig` makes the appropriate changes to the kernel and reboots the new kernel. This lets you install a slot card in any slot; `autoconfig` automatically makes the appropriate changes to the kernel.

Note that if the system contains a slot card for which the kernel does not contain a driver, `autoconfig` does not build a new kernel. In this case, `autoconfig` prints a warning message and continues. This allows you to add the appropriate software driver at a later time.

3. Changing the startup device and application

If you have more than one hard disk, you have the option of configuring any one of them to boot the system at startup time. For instance, you might want to change the startup device if one hard disk is devoted to the Macintosh Operating System and another is devoted to A/UX.

Note: If an A/UX disk is to be the startup device, it must contain `sash` and a Macintosh system folder.

Much as you can set the operating system that is automatically booted, you can set which application the system automatically runs upon system startup. Any Macintosh application can be the startup application, including `sash`. The capability to boot automatically into the application of choice is provided for convenience, allowing a user simply to press the Power On key to have his or her favorite application open on screen.

3.1 Steps to change the startup device

Initially, A/UX is configured to boot from the hard disk with the highest SCSI ID number. You can change this to have the system boot from any device, from SCSI ID 0 through 6. The procedure to change the startup device involves two main steps.

1. Selecting the correct device on the Control Panel Desk Accessory to let the system know where to start.
2. Identifying the startup device on the Preferences menu, a menu available from `sash`, by entering the SCSI ID number.

The procedure to carry out these steps is as follows:

1. Turn on the computer and enter the Macintosh Operating System. This means canceling the A/UX boot if the A/UX automatic startup screen appears.
2. Open the Control Panel desk accessory.

You reach the desk accessories from the Apple® symbol in the upper left of the screen.

Your system will now boot from the disk configured as the startup device.

3.2 Technical details

The steps the Macintosh follows when looking for startup files and the technical details of the algorithm used to boot A/UX from a disk are given in the following section for those who need this information.

3.2.1 The natural order of startup devices

When the computer starts up, it looks for a Macintosh system folder on each device, in a specific order and in specific places. This information may be of interest when you consider changing the Startup Device. The computer looks in the following places in the following order:

1. Internal 3.5-inch disk drive number 0
2. Internal 3.5-inch disk drive number 1
3. The internal hard disk
4. The hard disk with the highest SCSI ID number

You can override the natural order by setting the hard disk from which you want to start as the startup device.

3.2.2 How A/UX boots from a hard disk

A/UX is designed to be booted from any SCSI disk, from SCSI ID 0 through SCSI ID 6. The A/UX kernel accepts boot device information from `sash`. This design is unlike many other UNIX® kernels, which frequently have the boot device hardwired into the kernel and require use of `adb` or similar programs to patch in boot device changes.

The A/UX kernel requires two pieces of information about the disk that it will be running from: the root file system and the swap area. The A/UX kernel does not know about disks in general. Rather, it knows that it can call one of several block device drivers (by using the major

number to select one of the block device drivers), and that it can pass to that block device driver a parameter (the minor number) indicating just where to look.

When the `sash` application is launched, it finds the A/UX kernel (from one of several possible places) and loads it into main memory. It leaves in main memory a set of major and minor numbers for A/UX to use for the root file system and swap area. You can change these major and minor numbers by pulling down the Preferences menu and choosing General from `sash`.

3.3 Changing the startup application

The startup application is the program that appears on screen after you turn on the computer. Having a startup application is a feature of the Macintosh OS, so you choose a startup application from the Macintosh Finder. You can make any application your startup application. For example, if you use MacDraw® in most of your sessions with the computer, it makes sense to set MacDraw as the startup application.

By default, `sash` is the startup application on A/UX systems. As such, `sash` causes the A/UX automatic startup screen to display as part of the default booting sequence.

You can change the startup application at any time. Follow these steps to do so:

1. In the Macintosh Operating System, with the Macintosh Finder displayed, select the application you want to make the startup application.

This application will now be listed as an option when you complete the next step.

2. Choose Set Startup... from the Special menu.

The dialog box lists the options for the startup application.

Note: In the dialog box, both Finder and MultiFinder are shown as selectable options. However, of the two, Finder is more compatible with A/UX.

3. Click OK to accept the selected application as the startup application, or click Cancel if you want to reconsider your choice

and leave the startup application as it was before you opened this dialog box.

4. Select Restart from the Execute menu for the changes to take effect.

4. Setting system-specific details

You can set the following for your system:

- the time zone
- the name of your system
- the message of the day
- the mode your system automatically boots into (single- or multi-user mode)

Setting the time is not a mere detail. You must set the time if you are affected by Daylight Savings Time or if you move your machine to a different time zone.

4.1 Setting the system time

Both the A/UX and Macintosh operating systems normally keep track of the correct time without intervention. However, you need to intervene if one of the following events occur:

- Daylight Savings Time begins or ends.
- You move the system to a new time zone.
- A user logs in from a different time zone.
- A hardware problem modifies the system clock.

4.1.1 Time in A/UX and the Macintosh OS

The two operating systems keep the time quite differently but share a single hardware clock. You can set the time either in A/UX (with the `date` command) or in the Macintosh OS (on the Control Panel) and have it take effect in either operating system. However, adjustments for Daylight Savings Time need to be entered in both operating systems. A/UX stores the local time as an offset from Greenwich Mean Time (GMT) and so has the facility to adjust automatically for Daylight Savings Time. However, the Macintosh Operating System (OS) stores "wrist watch" time and so cannot adjust automatically when Daylight

Savings Time begins or ends. Since the two share a single hardware clock, the sophisticated means by which A/UX keeps time and adjusts for Daylight Savings Time is lost by the Macintosh OS's simplistic approach.

It is important that the A/UX system have the correct time. A/UX uses the time and date to monitor or control a number of activities. Specifically, without the correct time, A/UX cannot properly:

- log system events, such as mail activity and logins
- record file activity, such as the creation, modification, or accessing of a file
- track file status with utilities such as `make`
- schedule system and user tasks, such as removing core files and making file backups

A/UX stores the date and time as an offset of Greenwich Mean Time. This offset is called the GMT bias. When you give the `date` command to display the date and time, A/UX calculates your local time by using the GMT bias. You need to tell A/UX the GMT bias for your time zone.

As the system administrator, you must inform *both* operating systems of the correct time so that they maintain the correct date and time. In general, these are the procedures. Step-by-step instructions are contained in the following sections.

1. First, inform the Macintosh OS of the local time. This sets the hardware clock shared by the two operating systems.
2. Next, give the `sash` utility your GMT bias, which is the offset in minutes between Greenwich Mean Time and your time zone. (Figure 2-3 is a map with the GMT biases for the various time zones.) This allows A/UX to calculate the correct GMT values from the local time you supplied in the first procedure.
3. Last, tell the A/UX system which time zone to use for the system, and if Daylight Savings Time must be observed.

this page is also a place holder

this page is also a place holder

4.1.2 Resetting after a hardware problem.

If a hardware problem fouls up the hardware clock, you need to reset the time from either operating system. To reset the clock from A/UX, use the `date` command. For example, reset the clock to May 15, 1989 at 11:59 A.M., by entering

```
date 0515115989
```

For more information, see `date(1)` in *A/UX Command Reference*.

Be sure to enter the local date and time; A/UX can then calculate the GMT value for its internal use.

4.1.3 Resetting after moving a system to a different time zone

If you move the computer to a different time zone, you need to adjust the GMT bias. This is done from the Preferences menu, as follows:

1. Open `sash`. You do this by clicking Cancel when the A/UX automatic startup screen appears.
2. Choose General from the Preferences menu.
3. Enter the GMT bias, which is the number of minutes between your time zone and Greenwich Mean Time. Figure 2-3 shows the GMT biases around the world. However, if Daylight Savings Time is in effect, you must adjust the bias accordingly. For example, the eastern United States has a bias of -300 during Easter Standard Time, and -240 during Daylight Savings Time.

Now you need to inform A/UX of your time zone.

4. Close the General dialog box and launch A/UX by selecting Launch from the Execute menu or by typing `autolaunch` at the `sash` prompt.

After the boot is completed, inform A/UX of your time zone by modifying the files `/etc/zoneinfo/localtime` and `/etc/TIMEZONE`, as described in the next step.

5. To change the time zone, begin by entering

```
cd /etc/zoneinfo
```

To examine the list of time zones, enter

```
ls -CF
```

In response, a list of time zone files appears. The time zones are named in various ways, some by common abbreviations and others by country name. The four time zones in the continental United States are also shown with a notation such as PST8PDT. In addition, all time zones are listed as the number of hours off of Greenwich Mean Time. For example, the time zone for the eastern United States can be entered as US/East, EST5EDT, and GMT-5 during Eastern Standard Time and GMT-4 during Daylight Savings Time.

Some time zone listings automatically allow A/UX to manage Daylight Savings Time. They are the US time zones with notations such as EST8EDT, and the ones in the US directory (except, of course, the time zones in that directory that do not observe Daylight Savings Time). If you enter one of these time zone names and you do not care whether the clock in the Macintosh OS maintains correct time but only that the time in A/UX is correct, then entering one of these time zone names will save you from making adjustments for Daylight Savings Time.

Many time zones are listed as part of a directory, for example, Australia/. You list the subchoices in a directory by entering, for example,

```
ls -CF Australia
```

Make a note of the name of your time zone file, for example, Japan or Australia/Tasmania, so you can enter it in a following step.

6. Remove the file localtime (which is really a link to a time zone information file) so that a new one can be made. Type

```
rm localtime
```

Now create the new localtime link, and specify your time zone file. For example, if your time zone is Tasmania in Australia, type

ln Australia/Tasmania localtime

7. Now open the file `/etc/TIMEZONE` and add the name of your time zone. This is the name of a time zone you made note of in a previous step.
8. Finally, to have this change take effect, exit and log in again.

The time zone is now set. Check the result by entering

`date`

This command should cause the proper date and time to be displayed. If it does not, check that you chose the correct time zone and that the Control Panel in the Macintosh OS shows the correct time.

4.1.4 Resetting for Daylight Savings Time

Because of the shared hardware clock, you need to reset the time when Daylight Savings Time begins or ends. Whereas A/UX is capable of handling changes to and from Daylight Savings Time, the Macintosh OS is not. Consequently, when Daylight Savings Time begins or ends, you need to reset the time to have both operating systems display the correct time.

1. Reset the GMT bias on the Preferences menu in `sash`.

See "Resetting After Moving a System to a Different Time Zone" for the complete steps to change the GMT bias.

2. Reset the time in the Macintosh OS on the Control Panel desk accessory.

This prevents the Macintosh OS from changing the time back after you changed the GMT bias in the Preferences menu. See "Resetting After a Hardware Problem" for the complete steps to change the clock on the Control Panel.

3. Reboot the system to have the newly entered time take effect.

Resetting the time when Daylight Savings Time begins or ends is unnecessary in one situation: if your time zone file accounts for Daylight Savings Time automatically, and you don't mind that the Macintosh OS shows incorrect time. See "Resetting After Moving a

System to a Different Time Zone'' for a description of the time zone files that account for Daylight Savings Time automatically.

4.1.5 Overriding the default time zone

If a user wishes to display a time and date different from that used by the system, this can be easily accomplished by changing the TZ environment variable. A common reason for wanting to do this is logging in from a terminal in a different time zone, and thus wanting the system to use the time and date of that time zone. This is accomplished slightly differently depending on the command shell the user works in. For example, a user logging in from Iceland and working in the C shell, which normally stores environment variables in the user's .login file would enter

```
setenv TZ Iceland
```

A user logging in from Iceland and working in the Bourne or Korn shells, which normally store environment variables in the user's

```
TZ=Iceland
```

4.2 Changing the message of the day

If you log in as the root user, the /etc/profile system startup script executes several commands, including

```
cat /etc/motd
```

In response, the system displays a message contained in the file /etc/motd (for "message of the day"). In the standard A/UX distribution, this file contains

```
*****
*
*           W E L C O M E   T O   A / U X
*
*****
```

You can change the contents of /etc/motd to anything you like by editing this file with a text editor. The new text you enter will take the place of the Welcome to A/UX message-next time you bring the system down to single-user mode, reboot, or log in.

4.3 Renaming the system

To assign a name to your system or to change the system's name, use a text editor to open the `/etc/HOSTNAME` file. This file should contain two fields separated by spaces or a tab. The name of your system is the word in the first field of this file. Change the first field and save the file. This change will take effect the next time you shut the system down and reboot. See "Shutting Down the A/UX System."

4.4 About single-user mode

Single-user mode is one of several run levels available on the A/UX system. As the name implies, in single-user mode, only one person has access to the computer because the terminal ports are disabled. This quiet state of the machine is necessary for the performance of administrative tasks, such as checking the file system.

In single-user mode, you have all the privileges of the root user. You can create or destroy anything on the system—files, directories, or processes—in just a few keystrokes. This level of power is necessary to perform most administrative tasks, but you need to use it selectively. As shipped from the factory, A/UX is configured to boot into single-user mode automatically in anticipation of your needing this run level to take care of initial administrative tasks, such as adding user accounts and peripheral devices. But unless you need the power and privilege of single-user mode, run A/UX in multi-user mode. This good habit will decrease the potential of your making a mistake that damages the system and deters system abusers from sneaking in and wreaking havoc on the system.

4.5 Entering multi-user mode

The general run level for A/UX is multi-user mode. This run level is recommended even if you are the only user on a machine, because it guards against inadvertent system damage. In multi-user mode, most daemons are enabled, as well as all ports designated in the `/etc/inittab` file to permit user logins (see "Initial Processes: `/etc/inittab`," in this chapter).

When you start multi-user mode, the system performs the commands stored in the files `/etc/bchecker`, `/etc/brc`, and `/etc/rc`. In general, low-level processes such as daemons are started when the system enters multi-user mode.

You can manually move into multi-user mode when your tasks are completed by entering the command

```
init 2
```

Or, you can change the default setting so that the system moves into multi-user mode automatically upon system startup. Both of these ways to move from single- to multi-user mode are described in the next two sections.

If your system has one or just a few users, the two basic run levels, single-user and multi-user, will probably suffice for normal operation. If you have a system with many users and perhaps many modems, you may be interested in A/UX's more complex run-level capabilities, described in this chapter under Changing Run Levels.

4.5.1 Entering multi-user mode manually

1. To enter multi-user mode from single-user mode, enter

```
init 2
```

You will now see

```
INIT: New run level: 2
```

```
Do you want to check the file systems? (y or n)
[default:n]
```

You are asked if you want to check the file systems. Currently, the standard A/UX distribution provides only one user-accessible file system, which is the root file system. If you have not added more file systems (for example, on an external hard disk or 3.5-inch disk) you can press RETURN at this prompt, and A/UX will proceed without performing additional file system checks.

As the system enters multi-user mode, commands in /etc/inittab, /etc/rc, /etc/brc, and /etc/bcheckrc are executed (see "Initial Processes: /etc/inittab," later in this chapter). When this process is complete, you should again see the message of the day followed by the login banner:

Apple Computer, Inc. A/UX

login:

2. The system is now running in multi-user mode, and you may log in. After you log in, you see

TERM = (mac2)

3. Press RETURN to set your terminal type to the Macintosh II.

4.5.2 Entering multi-user mode automatically at boot time

The system defaults to single-user mode because this is most useful when you are first working with A/UX, looking around the system, and setting things up. After you have set up your system, you might wish to bypass single-user mode and automatically boot into multi-user mode.

To boot A/UX into multi-user mode automatically, edit the file /etc/inittab. Change the line

```
is:s:initdefault
```

to

```
is:2:initdefault
```

When you next boot the system it will proceed as in the default booting sequence and then you'll see the prompt:

```
Do you want to check the file systems? (y or n)
[default: n]
```

As distributed, A/UX has one file system, root. If you have not created other file systems, press `n` in response to this prompt. (If you press `y`, `fsck` repeats a check of the root file system.) If you have file systems other than root and want `fsck` to check those file systems, press `y` in answer to this prompt. For `fsck` to carry out this check, a few conditions must be met. The steps to meet those conditions are provided in "Multiple File Systems and `fsck`" in Chapter 8.

To have the system automatically check the file systems 5-seconds after entering multi-user mode, rather than prompting you about it, edit the files /etc/bcheckrc and /etc/sysinitrc. In both files, change the line

```
reply="`/etc/query`"
```

to

```
reply="`/etc/query -t 5`"
```

When you next boot the system, it should come up in multi-user mode and display this prompt:

```
Apple Computer, Inc.  A/UX
login:
```

4.6 Changing kernel parameters

As shipped from the factory, A/UX is configured optimally for a system with 2 megabytes (MB) of main memory. If your system contains more than 2 MB of memory, you might be able to improve performance by changing some kernel parameters. If your system has 2 MB of main memory, skip this section, because changing the kernel parameters on such systems is not recommended.

4.6.1 Systems with 4 or more megabytes of main memory

There are two reasons to change kernel parameters to improve performance. The first is to take advantage of your system's abundance of main memory by increasing the RAM cache. This is accomplished by increasing the NBUF parameter. The second reason is a need to exceed the kernel-configured limits on your system. Such limits are evident from the error messages that appear. It is best not to change kernel parameters (except NBUF to increase the RAM cache) unless you see one of these screen messages:

Error Message	Parameter Requiring an Increase
m_expand returning 0	NBUF
inode: table is full	NINODE
file: file table is full	NFILE
proc: table is full	NPROC

To improve performance on a system with 4 MB or 5 MB of main memory, do the following. If you aren't already in single-user mode, get there by running the `shutdown` program. To do so, bring the system into single-user mode by entering `init 1` or `init s`. Then enter the following commands to reset these kernel parameters and

have `autoconfig` build them into a new kernel:

```
kconfig -n /newunix
NBUF=1000
NINODE=100
NFILE=100
NPROC=100
autoconfig -IuS /etc/startup
Control-d
sync
sync
sync
reboot
```

NBUF should not exceed 1000 unless you have 8 MB of main memory, in which case you can increase NBUF to 2500. It is safe to double other parameters (NINODE, NPROC, and NFILE), each time you run into the kernel-configured limits that initially caused you to increase these parameters.

You must reboot to put the new parameters into effect because `kconfig` changes the kernel file and not the currently running kernel. See `kconfig(1)` for an explanation of these parameters.

4.7 Initial processes: `/etc/inittab`

The system's topmost program is `init` ("initial process"). It is the first process to run after you boot the system. The `init` program makes shells and spawns processes. It gets all its process information from the file `/etc/inittab`.

The first command `init` executes is the `/etc/sysinitrc` shell program, which performs such basic functions as setting the system's clock and running `/etc/fsck` before you see the single-user mode shell prompt.

The second line in `/etc/inittab` specifies the default initial run level:

```
is:s:initdefault:      #First Init State
```

The run level in the `/etc/inittab` file is specified in the second field of each entry, in this case `s` for "single-user." Once the initial run level is determined, `init` processes only those `/etc/inittab`

entries whose run-level field is the same as the run level currently in effect. When you enter multi-user mode by typing

```
init 2
```

you invoke the `init` process with a run level of 2. In this case, `init` reads `/etc/inittab` and invokes every command that specifies a run level of 2.

If you check the contents of `/etc/inittab` on your system, you will see the processes invoked by `init`. For example, the following processes are in `/etc/inittab`. (However, the process names are preceded by an *id* and a *run-level*, as discussed in the next section.)

`/etc/bcheckrc`

A startup script that runs `fsck` on the remaining file systems.

`/etc/brc`

A startup script that sets the permissions on pseudo-ttys.

`/etc/rc`

A startup script that mounts the file systems (if applicable) and performs some general housekeeping.

`/etc/getty`

A process that enables logins on serial ports. See Chapter 3, "User and Group Administration," and Chapter 7, "Managing Other Peripheral Devices."

4.7.1 `/etc/inittab` entry format

Each line in `/etc/inittab` is an entry containing four fields separated by colons and followed by an optional number sign (`#`) and an optional comment. The format of each entry is

id:run-level:action:command

where

<i>id</i>	unique entry identification
<i>run-level</i>	run level where the entry is processed
<i>action</i>	action to be taken with next field
<i>command</i>	command to be executed

The following two lines are in the `/etc/inittab` file distributed with the standard A/UX system:


```
co::respawn:/etc/getty console co_9600
...
00:2:off:/etc/getty tty0 at_9600
```

These lines are followed, respectively, by these comments:

```
# Console port
...
# Port tty0 (modem), set to "respawn"
```

In `/etc/inittab`, comments about the line are preceded by a number sign (#). These comments indicate that the first line above is the console port. Because the *run-level* field is empty, the console port is enabled at all run levels, including single-user. The second line is the modem port, `tty0`. If you change the *action* field from `off` to `respawn`, this port will be enabled in multi-user mode. This is explained in more detail in the following discussion of the *action* field.

The *id* field is an arbitrary identifier of one to four characters that makes the entry unique. Although the identifier is arbitrary, convention dictates that the identifier for an entry that affects a port be the corresponding port number. In the *id* field of the first of the preceding lines, `co` refers to the console port.

The *run-level* field tells `init` whether to process the entry. The *run-level* field can be any number from 0 to 6 (2, for multi-user, is the most common) or the character `S` or `s` (for single-user). If the *run-level* field is empty, `init` processes the entry at all run levels.

The *action* field specifies how to execute the *command* field if the entry's *run-level* field matches the system's run level.

respawn

Specifies the system to run this command when the designated run level is entered. If the command terminates for any reason, the system should run it again. If you specify `respawn` as the *run-level* field, the command runs whenever the run level matches the *run-level* of the entry. If you leave the *run-level* field empty, the process runs at all times and all run levels.

off

Specifies the system to turn off this command for this run level.

This ensures that the command does not run as long as the specified run level matches the *run-level* field of the entry in question.

once

Specifies the system to turn on this command for this run level. If the command is not executed for any reason, no further action is taken.

wait

Specifies the system to wait until the current command is completed before *init* goes to the next line. The *wait* parameter is very important when you mount devices or file systems that require the current command to be completed before the next is begun.

The fourth field is the command to be executed. When the run level of each of the entries matches the default run level, the command named in the fourth field is executed. In the preceding example, the command */etc/getty* would be executed for each entry, as modified by the *action* field.

4.8 Changing run levels: *init*

Once the system is started and running at the default run level, you can change the run level. Log in as the root user and enter

```
init run-level
```

where *run-level* is an argument to *init* that may be either a value from 0 to 6 or the letters *s* or *S*. See *init(1M)* in *A/UX System Administrator's Reference*.

When you enter this command, *init* (or *telinit*, because the two commands are linked) scans */etc/inittab* again and activates those entries whose *run-level* value is the same as that of the new run level, leaving all other processes untouched.

For instance, if the following lines appear in */etc/inittab*

```
01:23:respawn:/etc/getty tty1 at_9600
02:2:respawn:/etc/getty tty2 at_9600
```

both ports *tty1* and *tty2* are running a *getty* command when the system is at run level 2. But if you later type

`init 3`

the `getty` running to `tty2` is terminated, whereas the one running to `tty1` continues because both run level 2 and run level 3 are specified for that port.

The only way to modify how a process runs at a new run level is to specify it in `/etc/inittab`. In theory, you should specify every process that affects, for instance, a port, with reference to every single possible run level. In practice, you do that only for those processes that you may want to turn on and off for certain specific run levels. See Chapter 7, "Managing Other Peripheral Devices," for examples.

If you introduce a change in `/etc/inittab`, enter

`init Q`

to make the change known to the system without restarting it. This command forces `init` to reread `/etc/inittab`. However, some of the processes controlled by `init` spawn child processes, and these child processes are not terminated by the `init Q` command. You can terminate these processes individually by specifying their process ID number to the `kill` command, or by shutting down the system and rebooting.

The general functioning of your system, then, is determined by the current run level. The current run level is determined at startup by the default entry in `/etc/inittab`, and it can be changed via the `init` command. The current run level determines which commands (in the *command* field of `/etc/inittab` entries) are activated.

When running `init`, you should be careful not to bring about any changes that would disrupt the activities of users currently logged in. For instance, `init` could disrupt an operating modem if a `getty` command were sent to that port. Know your run states and make sure that changing them will not disrupt user activity.

5. Shutting down the A/UX system

The aim of shutdown is to bring the system to an almost inactive status before you turn off its power. No files should be open, no processes should be running in the background, and everything should be written to disk. Only then can you turn the power off to the system

with no danger of corrupting the file system.

The shutdown program automatically does the following:

- brings the system down to single-user mode (`init s`)
- unmounts the file systems, if you have multiple file systems (for example, if you have an external hard disk)
- terminates all processes (`killall`)
- flushes the file system buffers (`sync`)

5.1 Using the shutdown program

The shutdown program automates all the preceding steps to bring down A/UX and warns all other users on the system that the system is going down.

To use shutdown:

1. Log in as the root user on the system console. (Because the shutdown program moves the system from multi- to single-user mode, it must be run from the system console rather than another terminal.)
2. Enter

```
cd /
shutdown
```

The system responds by printing a banner and the date

```
SHUTDOWN PROGRAM
Wed Oct 28 12:11:50 PST 1987
```

followed by a shutdown delay (the interval the system will wait before beginning the file system shutdown):

```
Do you wish to enter your own delay to shutdown?
Default is 1 minute (y or n):
```

3. If a one-minute delay is satisfactory, just press RETURN. If you want more time (for example, if important processes are still running or if you need to notify other users that you are shutting down the system) or less time (to proceed directly with the shutdown because no other users running are processes and no

background jobs remain undone) enter

y

In this case, you see this prompt:

Enter your delay in minutes (0-60)

(No warning messages will be sent if input is 0):

To proceed directly, type 0. To delay the shutdown, type another number. For example, for a two-minute delay enter

2

4. In this case, shutdown asks if want to send a message.

Do you wish to enter your own message (y or n):

To print the default message, enter n (no). The following message is broadcast to all users:

Broadcast message from root (console)

Wed Oct 28 12:21:46 PST 1987

SYSTEM BEING BROUGHT DOWN FOR SYSTEM MAINTENANCE

Minutes to SHUTDOWN :2

To enter your own message, enter y (yes); shutdown will prompt you for a message. Enter whatever text you want to send and then send an *eof* (usually CONTROL-D).

5. After waiting the specified time interval minus 30 seconds, shutdown prints

Broadcast message from root (console)

Wed Oct 28 12:25:11 PST 1987

SHUTDOWN IN 30 SECS

LAST CHANCE TO LOGOUT

6. After 30 seconds, the shutdown program performs the following actions:

- stops all daemons and kills all remaining processes
- executes sync a number of times to flush the write buffers
- unmounts the file systems

- executes sync again
- executes the `init s` command to bring the system to single-user mode

As these actions occur, you will see messages like these:

Mounted devices are:

`/dev/dsk/c0d0s0 on / type 5.2 (rw)`

Unmounting file systems

`/dev/dsk/c0d0s0 on / type 5.2 (rw)`

**** SYSCON CHANGED TO /dev/console ****
Killing all processes

INIT: New run level: S

INIT: SINGLE USER MODE

This is followed by the message of the day and

`TERM = (mac2)`

Press RETURN to set your terminal type to Macintosh II.

- ☐ The system is now running in single-user mode, and the shutdown program has exited. To reboot, enter

`reboot`

This starts the boot sequence over again.

- ☐ To turn off the computer, enter

`powerdown`

In this case, switch off any external disks.

5.2 Shutting down the A/UX system manually

The `shutdown` program provides the safest, easiest means to shut down A/UX, and, as such, is the recommended method. Shutting down A/UX by hand is the other option. This second option is not recommended because it is subject to problems caused by typing mistakes, deleted steps, or any other operator error.

To shut down the system by hand:

1. Log in as the root user (`root`, `rootcsh`, or `rootksh`).
2. If necessary, move to the root directory.
3. If necessary, warn your users that you're shutting down the system by entering the command

```
wall
enter your message here
eof (usually CONTROL-D)
```

4. Wait a few minutes, then bring the system to single-user mode by issuing the command

```
init s
```

You should see the message

```
****      SYSCON CHANGED TO /dev/console      ****
```

followed by the shell prompt. However, you should not type anything at this point. Wait until you see the screen message

```
INIT: New run level: s
INIT: SINGLE USER MODE
```

followed by the message of the day and

```
TERM = (mac2)
```

Press RETURN to set your terminal type to Macintosh II.

5. Terminate all remaining processes by entering the command

```
killall
```

6. Update the disk to include changes made in the buffer cache by entering the command line

```
sync; sync; sync
```

7. If you have an external hard disk or 3.5-inch disk with a mounted file system, enter the command

```
umount file-system
```

where *file-system* is the name of any file system other than the

root file system. If the unmounted file system is on a 3.5-inch disk, eject the diskette by typing

`eject`

as necessary.

8. Update the root file system with any final changes made in the buffer cache by entering the command line

`sync; sync; sync`

- ☐ Then, to reboot, enter

`reboot`

This starts the boot sequence over again.

- ☐ To turn off the computer, enter

`powerdown`

After giving the `powerdown` command, switch off any external disks.

5.3 If your system hangs

If your system hangs for some reason and you are unable to invoke a command, you can bring down the system by pressing the power switch at the rear of the unit, or by using the programmer's switch. This is not recommended because it can introduce inconsistencies in the file system. If you bring the system down this way, make sure to run `fsck` on the root file system when you boot up again (see Chapter 8, "Checking the A/UX File System: `fsck`".)

PLACE TAB
HERE

— USER AND GROUP
ADMINISTRATION

BACK OF TAB

Chapter 3

User and Group Administration

Contents

1. The user's working environment	1
1.1 Components of a user's environment	1
1.2 Files that determine a user's environment	4
1.2.1 The <code>/etc/passwd</code> file	4
1.2.2 The <code>/etc/group</code> file	5
1.2.3 The <code>profile</code> startup scripts	8
1.2.4 The <code>kshrc</code> startup script	8
1.2.5 The <code>cshrc</code> and <code>login</code> startup scripts	8
1.3 How A/UX establishes the environment	9
2. File permissions	10
2.1 File access permissions	10
2.2 Directory permissions	12
2.3 Modifying a file's permissions	13
2.3.1 Symbolic terms	13
2.3.2 Numeric terms	14
2.4 <code>umask</code> and file permissions	16
2.5 The administrator's role in assigning permissions	17
2.6 <code>setuid</code> and <code>setgid</code> commands	17
3. Adding a user	18
3.1 Planning the user's working environment	19
3.2 Specifying a user's working environment	22
4. Modifying a user's working environment	25
4.1 System file permissions	26
4.2 Moving a user	26
4.2.1 Moving a directory	26
4.2.2 Using <code>tar</code> to move a user across file systems	26

4.2.3 Using <code>cpio</code> to move a user across file systems	28
4.3 Changing a user's startup program	29
5. Removing a user	30
5.1 Gentle deletion	30
5.2 Backup and selective deletion	31
6. Troubleshooting	31

Figures

Figure 3-1. Access classes	11
--------------------------------------	----

Chapter 3

User and Group Administration

1. The user's working environment

Within the A/UX system, each user has a working environment that is analogous to an office worker's work space. Each user's working environment provides the following features:

- **A secure place to work:** When users want to work with the A/UX system, they must first enter their login names and secret passwords. After logging in, they are automatically located in their own home directories. This permits a private environment where each user has control over who has access to his or her work.
- **The ability to share tools and data:** The A/UX system also provides a mechanism by which users can share their work with other users. This is done mainly through the formation of groups of users with common tasks. Through the prudent use of groups, you can set up environments that permit the appropriate mix of security and sharing of resources.
- **The ability to customize:** Users can, in most cases, modify many features of their working environments, by making changes to specific files located in their home directories. This gives them the power to personalize their environment, just as workers can set up their own work space in an office. Although users often modify their own working environments, the system administrator is responsible for the initial setup and certain kinds of modifications.

This chapter describes procedures and concepts that assist you in establishing, modifying, and removing users' working environments.

1.1 Components of a user's environment

A number of factors determine the interpretation and operation of commands given by a user logged into an A/UX system. This manual

refers to the collection of these factors as the "user's working environment" or more simply as the "environment"; this is a broader use of this term than that defined, for instance, in `environ(5)`.

The following are the components of a user's working environment:

- **Permissions:** Every file in the A/UX system is associated with permissions (or modes) that determine who can do what with the file. Three kinds of action can be performed on files, each associated with a corresponding type of permission: write, read, and execute. These three types of permission can be set differently for three types of users: the owner of the file, users belonging to the same group as the owner of the file, and other users, regardless of group membership. Users can execute programs or have access to data files only if their user IDs or group IDs are set appropriately for each program and data file. See "File Permissions" and "System File Permissions."
- **Login name and password:** Before users can gain access to the system, they must enter their login name and password. These are defined in the `/etc/passwd` file; see "Files That Determine a User's Environment."
- **User ID:** Each user login name is associated with a unique user ID (uid) that identifies the user. This is defined in the `/etc/passwd` file; see "Files That Determine a User's Environment." When a user attempts to use a command on some data, this number is compared to the user ID associated with both the command and the data file. If there is a match in either case, the files are checked to see how the permissions are set for the owner of the file. If there is no match for either file, the user's group ID is compared to the group ID associated with the files.
- **Group ID:** Each user login name is associated with at least one group ID (gid). This is defined in the `/etc/passwd` file; see "Files That Determine a User's Environment." The group ID numbers indicate the groups to which the user belongs at the time of login. Group membership is a security feature that permits some users access to files, while denying this access to other users. This inclusion and exclusion can be total or partial.

- **Home directory:** Each login name is associated with a home directory; this is defined in the `/etc/passwd` file (see "Files That Determine a User's Environment"). When users log in to the system, their startup programs use the home directory as the home base for creating and using files. In addition, the user can modify special files residing in this directory to tailor his or her environment further.
- **Current directory:** This is the directory where the user is located at the present time. Every time the user changes to a new directory, the new directory becomes the current directory. When a user specifies a relative filename (a filename that does not start with a `/`), the current directory is searched for a file of that name.
- **Startup program:** After logging in, the user needs some way of communicating with the A/UX system. The startup program is the program that automatically greets the user after a successful login. This is defined in the `/etc/passwd` file; see "Files That Determine a User's Environment." The startup program is usually defined as the Bourne shell, C shell, or Korn shell. If it is not specifically assigned in the `/etc/passwd` file, the default startup program is the Bourne shell. You can use any of the three A/UX command interpreters as the startup program. Sometimes the startup program is a more restricted and restrictive program than the A/UX shells. See "Changing a User's Startup Program" for more information.
- **Other environment variables:** The user's shell provides a number of environment variables that can be assigned different values to alter the environment. Some environment variables are automatically assigned values from the `/etc/passwd` entry for each user; these include `LOGNAME` (login name), `HOME` (home directory), and `SHELL` (startup program). Other variables are assigned values at the shell prompt or in files such as `.profile` in each user's home directory. One such variable that affects the user's working environment is the `PATH` variable. The value assigned to a user's `PATH` variable determines which directories,

and in what order, the shell will search for the file corresponding to a command issued by the user. For other environment variables, see *A/UX User Interface*.

1.2 Files that determine a user's environment

The user's environment is normally established at login time from information stored in a number of system files. See "How A/UX Establishes the Environment," earlier in this chapter, for a description of the order in which this is done. This section describes each of these files along with its format and content.

1.2.1 The */etc/passwd* file

A user's working environment is specified largely by a single entry in the */etc/passwd* file. You must be logged in as the root user to modify this file.

The */etc/passwd* file distributed with A/UX has several administrative logins and a user login named *start*. See "Administrative Logins on the A/UX System" in Chapter 1 for a description of the administrative logins.

Each entry consists of one line with seven fields separated by colons. The form of an entry is

login-name:password:uid:gid:misc:home-directory:startup-program

where the fields are interpreted as follows:

login-name

The name the user must use when logging in. It must be unique in the */etc/passwd* file.

password

An encrypted version of the actual password the user must use when logging in. The encryption is done automatically when the password is first assigned and whenever it is changed. Having the password encrypted makes it possible to have the */etc/passwd* file open for reading by everybody.

uid A unique user ID number for each user.

gid The user's default group membership. If the user is not listed in the */etc/group* file (see the next section, "The */etc/group* File"), the user belongs by default to the group

whose number appears in the user's *gid* field in the */etc/passwd* file.

misc Miscellaneous information about the user, such as full name, address, and telephone number. For other uses of the *miscellaneous* field, as well as for the exact specifications of the *password* field, see *passwd(1)* in *A/UX Command Reference* and *passwd(4)* in *A/UX Programmer's Reference*.

home-directory

The name of an existing directory whose permissions, ownership, and group membership are such that the user can have access to it. Whenever the user logs in to the system, this is the directory where he or she is initially located.

startup-program

The name of an executable program, usually one of the A/UX shells, that permits the user to communicate with the A/UX system.

The following is a typical */etc/passwd* entry:

```
joe:AxhlmzGfp:56:100:Joe Doe:/users/joe:/bin/sh
```

1.2.2 The */etc/group* file

The *gid* field of a user's entry in the */etc/passwd* file establishes a single default group for the user. The */etc/group* file is used to establish multiple group memberships for a user.

The */etc/group* file contains entries with four fields separated by colons. The form of each entry is

group-name : password : gid : list

where the fields are interpreted as follows:

group-name

The group name. Group names are arbitrary, but by convention their meanings should be self-evident (for instance, *acctg* rather than *xyz24*).

password

An encrypted version of the group's password. Although the *password* field can be set for each group, it is common practice

to disable group password checking by filling the *password* field with the words VOID or NONE, or with asterisks (*). The reason for thus disabling password checking (as would otherwise occur, for example, when the *newgrp* command is given) is that in practice people feel freer giving away group passwords than individual passwords. Disabling the *password* field prevents this common security breach.

- gid* An arbitrary number closely associated with the group name. For each group ID there is only one group name, and vice versa. The actual group ID numbers that exist in the */etc/group* file are the only numbers that should be entered in the *gid* field of */etc/passwd* entries. The group ID entered for each user in */etc/passwd* should coincide with the group ownership of that user's home directory.
- list* A list of the login names of the members of the group. The login names must be separated by commas. Entering a user's login name in the group's *list* field is optional for those users who belong to only one group. In this case, their group membership is determined by the value assigned in the *gid* field of their */etc/passwd* entry.

The following is a typical */etc/group* partial listing:

```
acctg:****:101:paul, john, peter
legal:****:102:rose, paul, john, peter
```

By convention, group names are arbitrary though self-evident. The groups themselves, though, should not be arbitrary. As system administrator, you should divide users into groups according to their assigned activities, and you should allow users to belong to different groups only when there is some overlapping of activities. If a user belongs to too many groups, you should ask yourself whether the group distribution you have established is the best one.

As an example of the use of the */etc/group* file to establish multiple group memberships for a user, look again at the preceding partial listing of an */etc/group* file. You will notice that Paul is listed both in *acctg* and in *legal*. If the group ownership of his home directory is *acctg*, then every file he creates and every directory he makes below her home directory will also belong to *acctg*. For various

reasons, Paul might want some of the files he creates to belong to the group `legal`. There are two ways he can accomplish this:

- He can create a file having group ownership `acctg` and then change its group membership to `legal` after the fact.
- He can make a directory having group ownership `acctg` and then change the group membership of the directory to `legal`; every file created within that directory will also belong to `legal`.

In both cases, Paul can change a file's or a directory's group membership by giving the command `chgrp`. For example, if Paul creates a file named `suits` in his home directory (group membership `acctg`), a long listing of the file (`ls -l`) will show

```
-rw-rw-r-- 1 paul acctg 19900 Oct 3 17:51 suits
```

To change the file's group membership, Paul has to enter

```
chgrp legal suits
```

After this command, a long listing would show the new group:

```
-rw-rw-r-- 1 paul legal 19900 Oct 3 17:51 suits
```

Note that the file permissions are not changed by the `chgrp` command, but now they apply relative to the group to which the file belongs.

Note that the way A/UX handles groups is derived from the method used by the Berkeley version of the UNIX operating system, and this method differs from the way System V, Version 2.2 of the UNIX operating system handles groups. (In the System V version, the user is allowed to be in only one group at a time.) In A/UX, a user can be in a maximum of eight groups. The system administrator enters the groups to which a user belongs into the file `/etc/group`. To list one's group memberships, enter `groups`. If a user belongs to eight groups and temporarily needs to be in yet another group, enter `newgroup groupname`. This causes `groupname` to replace the first group listed in `/etc/group` for the duration of the login session. Note that a user's group membership is still restricted to eight groups. The `newgroup` command temporarily substitutes the new group in place of the first group in `/etc/group`.

Now suppose Paul wants to create a new file `court/notes`. Before creating the file, the system checks to ensure he has the appropriate permissions. First, his `uid` is checked. If his `uid` matches that of his parent directory, the file is created. This requirement would be met if Paul was trying to create the file `/users/paul/court/notes`. If his `uid` does not match the `uid` of the parent directory, the `gid` is checked. If the directory's group is a valid group for his account, the file can be created. Paul would satisfy this condition when creating the file `/legal/court/notes` (because Paul is a member of the group `legal`). If neither of these conditions is met, the permissions of the parent directory are checked to see if all users are permitted to create files. If any of these conditions are met, the file is created. The file is assigned Paul's `uid` and the `gid` of the parent directory.

1.2.3 The profile startup scripts

When a user logs in and the Bourne shell or Korn shell is specified as the startup program in `/etc/passwd`, the system automatically runs several shell scripts before giving the user a prompt. One of these scripts is the file `/etc/profile`. Like each `$HOME/.profile` file, this is a script of shell commands, which typically exports certain shell variables and sets a file creation mask. This file is readable by all users but cannot be modified by normal users.

The `/etc/profile` file, if it exists, is run *before* the file `.profile` in the user's home directory and thus serves as a default `.profile`. A user can override any actions performed in the execution of `/etc/profile` by including the appropriate commands in his or her own `.profile`.

1.2.4 The kshrc startup script

When a user logs in and the Korn shell is specified as the startup program in `/etc/passwd`, the system runs the `/etc/profile` and `$HOME/.profile` files described in the preceding section. If one of these files sets the `ENV` variable to any filename (`$HOME/.kshrc` in the standard A/UX distribution), the system also reads the contents of the named file.

1.2.5 The cshrc and login startup scripts

When a user logs in and the C shell is specified as the startup program in `/etc/passwd`, the system automatically runs several shell scripts before giving the user a prompt. One of these scripts is the file

`/etc/cshrc`. Like each `~/.cshrc` file, this is a script of shell commands, which typically exports certain shell variables and sets a file creation mask. This file is readable by all users but cannot be modified by normal users.

The `/etc/cshrc` file, if it exists, is run *before* the file `.cshrc` in the user's home directory and thus serves as a default `.cshrc`. A user can override any actions performed in the execution of `/etc/cshrc` by including the appropriate commands in his or her own `.cshrc`.

The `.login` script is run *after* `.cshrc`. It is typically used to set up terminal defaults and environment variables.

1.3 How A/UX establishes the environment

A close look at a successful login will help you understand how all the elements mentioned to this point interact in determining a user's environment.

1. Upon a successful login, the `login` program reads the user's *uid*, *gid*, and *home-directory* fields in the `/etc/passwd` file. Next it invokes the `initgroups` program, which reads each line of the `/etc/group` file, looking for a match between the user's login name and the *login-name* field in this file. For each match, `initgroups` assigns to the user the corresponding group specified in the *gid* field. The `login` program then executes the command named as the user's startup program in the *command* field of the `/etc/passwd` entry. This command inherits the user's user ID, group ID(s), and home directory from the `login` process.
2. Usually, the startup program is a shell and its initial invocation is known as the login shell. Login shells look for and (if it exists) read a startup file in the directory `/etc`. This file is known as the default startup file. If the startup program is the Bourne shell or the Korn shell, this file is called `/etc/profile`; if it is the C shell, the file is called `/etc/cshrc`. In this file, the system administrator can modify certain aspects of all the users' working environments, such as `PATH`, `HOME`, `TERM`, and `EXINIT`, as well as set other features such as aliases in the C

shell and functions in the Bourne shell. See `sh(1)`, `ksh(1)`, and `cs(1)` in *AUX Command Reference*.

3. After reading the default startup file, the shell looks for and (if it exists) reads an initializing file in the directory named in the *home-directory* field of the `/etc/passwd` file. If the startup program is the Bourne shell or the Korn shell, this file is called `.profile`; if it is the C shell, there are actually two files, `.cshrc` and `.login`. In this file, the user can modify certain aspects of his or her working environment, such as `PATH`, `HOME`, `TERM`, and `EXINIT`, as well as set other features such as aliases in the C shell and functions in the Bourne shell.
4. Once the startup environment has been thus established, the startup program prompts the user for input.

2. File permissions

Permissions determine who can and cannot have access to and use a particular file or directory. For example, if a file grants read-write permission to all users on the system, anyone on the system can get into the file and permanently change it. If a file grants read-only permission to all users on the system, all users can read the file, but no one except the user can make any changes to the file.

2.1 File access permissions

A user can set the following permissions on the files he or she owns:

- r** Read permission. Allows designated users to read a file or to copy its contents.
- w** Write permission. Allows designated users to modify a file.
- x** Execute permission. Allows designated users to execute a file (that is, to run it as a command).

Note that these permissions apply to regular files. For directory permissions, see the following section, "Directory Permissions."

These permissions can be set for users in three classifications called **access classes**. An access class is a category of user which may be *user*, *group*, or *other*. Each access class is represented by three characters; the order of characters is **r**, **w**, **x**, indicating the access permission granted to that category of user. If a hyphen appears

instead of an *r*, *w*, or *x*, permission to perform that action is denied to that category of user.

Ten characters are used to represent a file and its permissions; the first character indicates the type of file, and the next nine characters indicate the permissions of the three access classes as shown in Figure 3-1.

The file access permissions described below would appear on the screen as

```
-rwxrw-r--
```

-	rwx	rw-	r--
<i>type</i>	<i>user</i>	<i>group</i>	<i>other</i>

Figure 3-1. Access classes

- | | |
|--------------|--|
| <i>type</i> | The first character represents the file type. The type is not an access class, but type is an essential part of file permissions. In Figure 3-1, the file is a regular file (represented by -). Other file types are <i>d</i> (directory), <i>c</i> (character device), <i>b</i> (block device), <i>p</i> (FIFO or named pipe), <i>s</i> (socket), and <i>l</i> (symbolic link). |
| <i>user</i> | The next three characters in Figure 3-1, <i>rwx</i> , represent the file access permission of the owner of the file, known as the <i>user</i> . The owner has permission to read, write, or execute the file.) When used with <i>chmod</i> and <i>chgrp</i> , <i>user</i> is represented by <i>u</i> . See "Symbolic Terms." |
| <i>group</i> | The next three characters in Figure 3-1, <i>rw-</i> , represent the <i>group's</i> permissions. any user in the same group as the owner of the file has permission to read and write (change) the file. Permission to execute the file is left as a hyphen, meaning that the file is not executable by the group. When used with <i>chmod</i> and <i>chgrp</i> , <i>group</i> is represented by <i>g</i> . See "Symbolic Terms." |

other The last three characters represent the permissions for all *other* system users. In Figure 3-1, only *r* appears, meaning that others (who do not belong to the group) can only read the contents of the file. When used with *chmod* and *chgrp*, *other* is represented by *o*. See "Symbolic Terms."

Permissions have to be set for the three access classes for each file. They can be modified manually at any time, or they can be set automatically to be the same every time a file is created.

2.2 Directory permissions

Directory permissions work a little differently from file permissions. Here is a list of directory access permissions and their meanings:

- r* Filenames can be read from the directory.
- w* Directory entries can be added or deleted.
- x* The directory can be searched or made the current directory.

When you set permissions on a directory, you define who may list that directory's contents, add or delete files in that directory, or move into that directory. Note that setting permissions on a directory affects only the directory itself and does not change the permissions settings of any of the directory's files or subdirectories.

Directory permissions are one of the most important and neglected aspects of the user's environment. For example, file permissions that protect against reading or writing by other users are not enough to protect the file from being deleted if the directory permissions allow other users write permission. Similarly, if the directory grants the group read permission, its files can be listed by a group member even if the files themselves deny group read permission.

Group membership is an important consideration for the administrator setting up directory permissions. The default group membership of a file or directory is the same as the group membership of the directory in which the file is created. This allows for the creation of hierarchies of directories according to their group membership.

Additionally, directory permissions can affect the accessing of a file. If a wildcard (such as '*' or '?') is used in the path specification, read permission will also be required for the affected directory. This is a result of the wildcard causing the shell to read the directory (on the user's behalf) to find the requested file. Removal of read permission from directories can thus be used to prevent snooping while allowing access to specific files.

2.3 Modifying a file's permissions

Only the owner of a file, or the superuser, can change that file's permissions. This is also called changing a file's mode. The command used for this purpose is `chmod`. Anything that `chmod` can do to a file's permissions it can do to a directory's permissions as well, because A/UX treats a directory as a file. See the preceding section, "Directory Permissions."

2.3.1 Symbolic terms

The `chmod` command can be invoked with either symbolic or numeric terms. Symbolic terms are straightforward: `u` stands for user (that is, owner) of the file, `g` stands for group, and `o` stands for others; `+` represents granting permission, and `-` represents denying permission.

The format for invoking `chmod` with symbolic terms is

`chmod access-class operator permission filename-list`

(Note that there are no spaces separating *access-class*, *operator*, and *permission*.)

The arguments are:

access-class

One or more of the three access classes—*user* (`u`), *group* (`g`), or *other* (`o`)—described under "File Access Permissions," earlier in this chapter. Also, the access class *all* (`a`) lets you grant or deny permissions to all three access classes simultaneously.

operator

Grant access permission (the `+` operator) or deny it (the `-` operator). You can't both grant and deny permissions in a single command. You must grant permissions to one access class in one command, then deny it to another access class in a second command.

permissions

Read permission (r), write permission (w), and execute permission (x). You can grant (or deny) more than one type of permission at the same time, but you can't grant *and* deny permission at the same time.

filename-list

The file(s) whose permissions are to be changed. You may use absolute or relative pathnames.

To change the permissions of a file from

`-rw-rw-rwx`

to

`-rwxrw-r--`

the sequence of commands is as follows.

1. To grant execute permission to the user:

`chmod u+x filename`

2. And then, to deny write and execute permissions to all others:

`chmod o-wx filename`

2.3.2 Numeric terms

Numeric or absolute terms are based on the combinations allowed by octal numbers where, for each access class, the mode of the file is set as follows:

0	grants no permission
1	grants execute permission
2	grants write permission
4	grants read permission

These numbers can in turn be combined in the following way:

3	(1 + 2) grants execute and write permission
5	(1 + 4) grants execute and read permission
6	(2 + 4) grants read and write permission
7	(1 + 2 + 4) grants all permissions

The format for invoking `chmod` with numeric terms is

```
chmod permission filename-list
```

where *permission* is a composite of the three access classes.

The file permissions

```
-rw-rw-rw-
```

are represented numerically as

```
666
```

Using numeric terms, you give this command to make the file readable, writeable, and executable by the owner and group, and inaccessible to others:

```
chmod 770 filename
```

The first 7 represents `rwX` for the user, the second 7 represents `rwX` for the group, and the 0 represents no access permission for all others. The permissions of the file would then be

```
-rwxrwx---
```

Turning on the set-id or set-gid bits is very useful with very specific and restricted files, for example, the `_passwd` program. You can read about the set-id and set-gid bits and how they are used in the following section, "setuid and setgid Commands."

The command to turn on the set-gid bit on a file with read, write, and execute permissions for all (mode 777) is

```
chmod 2777 filename
```

The command to turn on the set-id bit on a file with read, write, and execute permissions for the owner, read and execute permissions for the group, and no permissions for all others (mode 750) is

```
chmod 4750 filename
```

The permissions field in the output of the `ls -l` command in the first case is

```
-rwxrwsrwx
```

where the `s` in the group execution field represents the set-gid bit.

The permissions field in the output of the `ls -l` command in the second case is

```
-rwsr-x---
```

where the `s` in the owner execution field represents the set-id bit.

You can combine the setting of the set-id bit and the set-gid bit, as you can with all other numeric terms, so that

```
chmod 6755 filename
```

will result in

```
-rwsr-sr-x
```

2.4 `umask` and file permissions

The environment variable `umask` defines the permissions for each file created by a user. You can set this variable for all users in the `/etc/profile` or `/etc/cshrc` files, or you can set it individually for each user in his or her `.profile` or `.login` or `.cshrc` file. See "How A/UX Establishes the Environment." The value assigned to `umask` in the individual files `.profile`, `.login`, or `.cshrc` overrides the values set in `/etc/profile` or `/etc/cshrc`.

The `umask` variable, like the permissions associated with `chmod`, can be assigned a numeric value of three octal numbers. The value of each specified digit is subtracted from the corresponding digit specified by the system for the creation of files.

For example, to ensure that all files created by a user have the permissions

```
-rw-rw-rw-
```

you must set the `umask` for that user as

```
umask 111
```

so that, when the 111 is subtracted from 777, the files' permissions are 666.

Similarly, to ensure that all files created by a user have the permissions

```
-rw-r-----
```

you must set the `umask` for that user as

`umask 137`

so that, when 137 is subtracted from 777, the files' permissions are 640.

The notation

`umask 77`

is shorthand for

`umask 077`

That is, leading zeros can be eliminated from the notation.

Note that changing a user's `umask` does not affect the permissions on existing files.

2.5 The administrator's role in assigning permissions

The A/UX system administrator needs to provide adequate security for the users and projects on the machine. Initial security for accounts should thus prevent other users from reading or writing to a new user's area. Individual users may, of course, override this initial setup, but relaxation of security should be an option, not the default.

To do this, the administrator must set up appropriate entries in `/etc/passwd` and `/etc/group`. It is just as critical for the user and group ownerships and permissions on the user's home directory to be correct. Finally, it may be appropriate to set the `umask` in the system-wide shell initialization files.

When you set up common areas for group activity, it is often useful to create a user name for the project itself. Certain privileged users can then log in to that account and perform administrative tasks not allowed to all members of the project's group.

2.6 `setuid` and `setgid` commands

It is possible, under A/UX, to set up commands that act as if they were being invoked by a specified user, or by a member of a specified group. The mechanism for this is simple: a `setuid` command takes on the user ID of its owner (the owner of the file being executed). The `setgid` commands function similarly but take on the group ID of the executed file.

For example, a user might wish to change his or her password in the `/etc/passwd` file. It would normally be quite insecure to allow every user to modify the file in question, so a `setuid` program, `passwd`, is used. When invoked, `passwd` takes on the identity of the root user for the time needed to edit `/etc/passwd`.

Of the two, `setgid` commands tend to be safer, since group membership typically confers less power. Both should be treated with respect, however. Some variants of UNIX allow shell scripts to be made `setuid` or `setgid`. This is not a secure practice, though, and should generally be avoided. (There are many ways in which shell scripts can be made to act as "Trojan horses.")

In any case, it may be desirable to have a `setuid` program that can only be run by a selected set of users. This can be accomplished by putting the set of users into the same group to which the program belongs and giving execute permission to group members. These users can then run the program, performing the action as if they were the owner of the executable file.

You can use `chmod` to turn on the set-uid bit or set-gid bit for a file. You use the fourth field of the `chmod` command for this. The meaning of the numbers corresponding to this field is

- 1 set sticky bit (not used by A/UX)
- 2 set gid bit on execution
- 4 set uid bit on execution

In swapping systems, the sticky bit indicates that the file should remain in main memory once it has been loaded in; this can shorten initialization time for frequently used programs at the cost of tying up a portion of main memory indefinitely. Because A/UX is a paging system, however, the sticky bit has no effect. For additional information see Chapter 10, "System Activity Package." Note that neither set user ID or set group ID modes apply to directories or other nonexecutable files.

3. Adding a user

Adding a user to your system is a process with two distinct steps: planning the new user's working environment and then specifying it.

These steps are equally important. Skipping the planning stage can lead to a very inefficient use of the system.

3.1 Planning the user's working environment

If at this point you do not have an actual user to add to your system but want to practice, you can use the examples listed below. If you are about to add a real user, follow the steps below but provide your own specifications.

Before you begin, you should have a clear idea of who the user is, what his or her tasks will be, what group or groups are currently engaged in similar activities, what parts of the system you want the user to have access to, whether a new group should be created, and where in the system the new user should be located.

In other words, the new user should belong to a group whose members have similar tasks (accounting, legal, programming, documentation, and so on) or to more than one group if the user will have a variety of tasks.

Follow these steps in planning a user's working environment:

1. **Keep a hard-copy record of data about the new user.** The record should include information like that listed in the form below. This form simplifies adding a new user's working environment and is a useful record to keep.

User's real full name _____

Date (year/month/day) _____

User's telephone number _____

User's login name _____

User identification number _____

Group identification number _____

Group name _____

Full pathname of the user's home directory

Full pathname of the user's startup program

2. Pick a login name for the user. Login names normally consist of lowercase alphabetic characters only. To make sure that the new user's login name is a new name, enter the command

`grep login-name /etc/passwd`

This command searches the `/etc/passwd` file to see if a user already has the login name you have chosen. If you see a line starting with the login name you have chosen, pick another login name and invoke `grep` again with the new name. If you see only the shell prompt and no line starting with the name, no one is using that login name. Enter the new login name on the form in step 1. If you are practicing, enter the name `dummy`.

3. Before selecting a new user identification number, you must find one that is not being used. One method for selecting the lowest unused number is to enter the command

```
cut -f3 -d: /etc/passwd | sort -n
```

This displays the current user ID numbers in the `/etc/passwd` file. Pick a number that is not being used, and write this number in the space labeled "User identification number" on the form in step 1. By convention, ID numbers under 100 are reserved for special uses, such as for programs or system operators.

4. Select a group identification number. If you are practicing, use 100; otherwise, see "The `/etc/group` File," earlier in this chapter, for information about selecting and specifying group membership. Write the number in the space labeled "User's group identification number."
5. Select a home directory. Use `/dir/login-name`, where *dir* is the directory where you are going to put the new user accounts and *login-name* is the user's login name on the form. If you are practicing, use `/users/dummy`. Think carefully. You may want to use a pathname like

```
/dir/gm/login-name
```

where *gm* represents a directory above the user's home directory. This *gm* directory should have the same group membership as the user's home directory, but the user should not have write permission on it. All users belonging to the same group should then have their home directories at the same level, that is, under *gm*. This way, the owner of the *gm* directory can be the group manager. Who should be the group manager is also a matter for thought. Once you have decided, write down the full pathname in the home directory space on the form. If you have a second disk it might be useful to create a file system to hold user accounts.

6. Select a startup program, such as `/bin/sh` or `/bin/csh`. If you are practicing, choose `/bin/sh`. Otherwise, you may also ask for the user's preference. Write down the full pathname in the startup program space on the form. See "Changing a User's

Startup Program'' for information about using different command interpreters as a user's startup program.

3.2 Specifying a user's working environment

Now that you have made your choices and have written down all the information, you can proceed with the practical steps involved in adding the user.

1. If you are not already the superuser, log in as the root user.
2. Make a copy of `/etc/passwd`. For instance,

```
cp /etc/passwd /etc/passwd.old
```

This copy is your backup in case you accidentally destroy this critical file.

3. Next, use the `vipw` command to edit the `/etc/passwd` file. You should have all the pieces of information in front of you.

Note: The `/etc/passwd` file is set as "read-only." The `vipw` editor copies the contents of the password file into a temporary file (`/etc/ptmp`). After you edit and write the file, the editor copies the changes back to the `/etc/passwd` file. The `vipw` editor locks `passwd` so that it can't be edited with `passwd(1)` while the— copy is in use.

See `vipw(1M)` in *A/UX System Administrator's Reference* for more information about using `vipw` to edit `/etc/passwd`.

4. Enter the following line as the last line in the file, replacing each italicized word with the new user's information from the form you just completed. *Be careful while you modify this file.* It is essential to your users' and your own ability to gain access to the system.

*login-name : * : uid : gid : : home-directory : startup-program*

Enter `*` in the second field for now. It will be filled by an encrypted version of the user's password in a few moments. The fifth field, which is also blank in the preceding line, is for any miscellaneous information you care to enter (for example, the

user's real name if it differs from the login name, phone number and address, and so on). Remember to use full pathnames for the user's home directory and startup program. If you want to play it safe, enter the following line:

```
dummy::50:100:nice guy:/users/dummy:/bin/sh
```

5. Write the file and quit the editor.
6. Now enter the command

```
passwd login-name
```

where *login-name* is the name you filled in on the form in the space labeled "User's login name." You are asked to enter the new user's secret password. The `passwd` program asks you to enter the password twice. If you do not type the same password two times, it asks you to try again. If the word is too short (fewer than six characters), it asks you to enter a different word (see `passwd(1)`). Write the password down and give it only to the new user. The new user should log in and set a new password as soon as possible.

7. Create the user's home directory, using the full pathname from the form, with the command

```
mkdir home-directory
```

If you are practicing, enter

```
mkdir /users/dummy
```

8. If you have a standard `.profile` file, use this command to give the user a copy of it:

```
cp profile-standard home-directory/.profile
```

Replace the words *profile-standard* and *home-directory* with the name of the standard `.profile` file and the user's home directory as entered on the form. Note that the A/UX standard distribution supplies basic copies of suggested login and environment files needed for each of the A/UX shells. These are located in `/usr/adm`.

If the user's login shell is the Bourne shell (*sh(1)*), you can use the command

```
cp /usr/adm/sh.profile home-directory/.profile
```

If the user's login shell is the Korn shell (*ksh(1)*), use the commands

```
cp /usr/adm/ksh.profile home-directory/.profile  
cp /usr/adm/ksh.kshrc home-directory/.kshrc
```

If the user's login shell is the C shell (*csh(1)*), use the commands

```
cp /usr/adm/csh.login home-directory/.login  
cp /usr/adm/csh.cshrc home-directory/.cshrc
```

If you are practicing, use the *.profile* file from the start account (used in *Getting Started with A/UX*) with the command

```
cp /users/start/.profile /users/start/dummy
```

9. Now you can change the ownership of the user's home directory and login or environment file(s). Again, replace each of the italicized words with the information on the form. Enter the commands

```
chown login-name home-directory  
chown login-name home-directory/login-files
```

where *login-files* are the files you copied from */usr/adm*.

If you are practicing, change the ownership as follows:

```
chown dummy /users/dummy  
chown dummy /users/dummy/.profile
```

10. Next, change the group membership of the user's home directory by entering the command

```
chgrp group-name home-directory
```

where *group-name* is the name (as listed in */etc/group*) of the group ID specified in the *gid* field of the user's entry in the */etc/passwd* file.

If you are practicing, enter

```
chgrp project /users/dummy
```

11. Now use these commands to change the permissions associated with the user's home directory and `.profile` file:

```
chmod 750 home-directory  
chmod 640 home-directory/.profile
```

The `750` grants the user write, read, and execute permissions, grants members of the group read and execute permissions, and denies all permissions to other users. The number `640` grants the user write and read permissions, grants members of the group read permission, and denies all permissions to other users.

If you're practicing, change the permissions with the commands

```
chmod 740 /users/dummy  
chmod 640 /users/dummy/.profile
```

For more information about the command lines in steps 8, 9, and 10, see "File Permissions," earlier in this chapter, and `chown(1)` and `chmod(1)` in *A/UX Command Reference*.

12. Now log out and log back in using the new user's login name and password. Create a new file in the working environment you just established. If you have any trouble, see "Troubleshooting."

4. Modifying a user's working environment

A/UX provides great flexibility in establishing and modifying a user's working environment. The following are several of the more important parameters that can be modified:

- a particular user's ability to have access to the commands and data stored on the system
- the accessibility of a user's files and directories to other users
- the location or name of any user's home directory
- the command that the user employs as the shell

4.1 System file permissions

The system administrator can change the permissions of system command and data files so that no users, some users, or all users can have access to them. This is a responsibility that should be exercised with extreme caution, because giving write permission to all users on a file like `/etc/passwd` can have disastrous consequences.

Users can change the permissions associated with their own files. For a discussion on how to do this and what effects these changes have on users' ability to have access to the files, see "File Permissions," earlier in this chapter, and `chmod(1)` in *A/UX Command Reference*.

4.2 Moving a user

Sometimes it is necessary to move a user's working environment. There are a few ways of doing this, and the method you choose depends on the characteristics of the move. If you do move a user's files, remember to change his or her home directory in `/etc/passwd`.

4.2.1 Moving a directory

The simplest move is the one that involves moving a user's directory to another place in the same file system. The command line

```
mv old-dir new-dir
```

moves the *old-dir* directory (including all its files, any subdirectories associated with it, and all of their files) to *new-dir*.

4.2.2 Using `tar` to move a user across file systems

While `mv` works only within the current file system, the `tar` command can be used to copy directories from one file system to another.

Note: In the standard A/UX distribution on a Macintosh II with an 80 MB hard disk, the disk contains only one user-accessible file system. The entire A/UX directory hierarchy and any specific hierarchy (such as `/usr`) are available on the root file system.

If you have created a new file system (for example, named `/users2`) on an external hard disk or diskette, you can copy all files and subdirectories contained in the directory `/users` to a directory

`/users2/john` on the other file system. To do so, use the command

```
cd /users
tar cf - john | (cd /users2; tar xf -)
```

The parts of this command line are as follows:

`tar` Command name.

`c` Creates a new `tar` image.

`f` Stores the image under the filename (or directory name) that appears next in the line.

`-` When used with `f`, directs the image to the standard output.

`john`

Name of the directory to start copying from.

`|` Connects or "pipes" standard output of the previous command to standard input of the next command.

`(...)`

Parentheses enclose commands to be executed in a subshell.

`cd /users2`

Changes the subshell's current directory to `/users2`.

`;` Command separator.

`tar` Command name.

`x` Extracts file(s) from the just-created `tar` image on tape or disk. The `tar` command does so file by file; if the file is a directory, it is extracted recursively (that is, until it is exhausted of files and subdirectories).

`f` Extracts the image from the following file.

`-` When used with `f`, takes the image from the standard input. When `-` stands for a filename, `tar` uses the standard output as a file with the `x` or `t` options.

This moves a copy of the user's files to a new directory. You can delete the original files once you are sure that the move was successful.

Please keep in mind that this example shows how to move the user and is not a lesson on `tar`; see `tar(1)` in *A/UX Command Reference*. Remember that when you move the user's files you should also change the user's *home-directory* field in `/etc/passwd` and any other references to his or her home directory in files such as `.profile`.

4.2.3 Using `cpio` to move a user across file systems

With `cpio`, a directory containing files and subdirectories can be copied elsewhere on the system, with all files maintaining their original ownership, permissions, and modification time.

Note: In the standard A/UX distribution on a Macintosh II with an 80 MB hard disk, the disk contains only one user-accessible file system. The entire A/UX directory hierarchy and any specific hierarchy (such as `/usr`) are available on the root file system.

If you have created a new file system (for example, named `/users2`) on an external hard disk or diskette, you can copy all files (except for dot files) and subdirectories contained in the directory `/users/john` to a directory `/users2/john` on the other file system. To do so, use the command

```
cd /users
find john -depth -print | cpio -pdm /users2
```

The parts of this command line are as follows:

`find`

Name of the command that gathers the filenames to pass to `cpio`.

`john`

Name of the directory to start the search from.

`-depth`

Forces a depth-first search of the directory, to control the order in which files are copied.

`-print`

Prints each file or directory name found.

- | Connects standard output of the previous command to standard input of the next command.

cpio

Name of the command that does the actual copying.

- Character signaling that options follow.
- p Copies ("passes") the named files to a named directory.
- d Creates new subdirectories as needed.
- m Retains the original file's modification times.

/users2

New directory in which to place the files.

This moves a copy of the user's files to a new directory. You can delete the original files once you are sure that the move was successful.

Please keep in mind that this example shows how to move the user and is not a lesson on **cpio**; see **cpio(1)** in *A/UX Command Reference*. Remember also that when you move the user's files you should also change the user's *home-directory* field in **/etc/passwd** and any other references to his or her home directory in files such as **.profile**.

4.3 Changing a user's startup program

The last field in the **/etc/passwd** file determines a user's startup program. Typically the field is **/bin/sh**, **/bin/ksh**, or **/bin/csh** (for the Bourne shell, the C shell, or the Korn shell, respectively). To change a user's startup program, all you have to do is change this field. Other modifications to the user's working environment may be necessary, however, particularly with regard to shell startup files in the user's home directory; see "Files That Determine a User's Environment," earlier in this chapter.

Any program at all can serve as the startup program. For instance, the last field of the **/etc/passwd** file can be a program such as **who**. If **who** is the startup program, the user is able to log in but sees only the output of the program **who** before being logged out, without ever getting a shell. Although **who** is not a very useful working environment, other programs, such as **rsh(1)**, are. The **rsh** program

allows a user full use of a shell within the home directory but allows no directory changes.

5. Removing a user

Removing a user from your system may be as simple as inserting a word such as `VOID` in that user's encrypted password field in `/etc/passwd`. However, if the user has created many files that must be saved, you may need to find all files owned by the user, back them up, examine each of them, determine who else uses the files, change the ownership of shared files, remove links, and finally delete the user's password entry.

This section introduces the most moderate form of user removal first and then discusses additional steps that make the removal more extreme.

5.1 Gentle deletion

The first step in removing a user from your system is to deny the user access to it. The cleanest way to do this is to edit the user's `/etc/passwd` entry and enter the word `VOID` in the encrypted password field. This makes it impossible for anyone to log in as that user (although that user's files remain unaffected).

Note: Do not leave the *password* field blank. A blank password is a serious security breach because anybody can then log in to the system with that login name and no password.

Do not delete the whole `/etc/passwd` entry for that user yet. If you do, you will not only deny the user access to the system but also affect the files owned by that user. Commands that use login names as arguments (for example, `chown` and `find`) or print information relating to login names (for example, `ls -l`) check the `/etc/passwd` file for the user names and/or numbers. If there is no login name for a file's owner, it is replaced by a number (when you enter `ls -l`, for instance). If you delete a few `/etc/passwd` entries, you will probably get confused about which files belong to which ex-user.

5.2 Backup and selective deletion

You need to be careful when deleting a user's files. In general, it is a good idea to back up a user's files before deleting them, for two reasons:

- These files may contain information that you may need later.
- These files may be used by other users on your system.

To locate all the files owned by the user and not below his or her home directory, follow these steps:

1. Void the user's password; see the preceding section, "Gentle Deletion."
2. Find all the files belonging to the user, regardless of their location, with the command

```
find / -user login-name -print > somefile
```
3. Back up the files using either `tar` or `cpio`; see the information on partial backups in Chapter 4.
4. Delete the user's files. Before you do so, it is a good idea to find out if anyone is currently executing any commands or using any data files owned by that user. Inquire personally or through mail, or use the `acctcom` command (see Chapter 9, "System Accounting Package") to see whether any others regularly use files created by that user. If they do, change the ownership of those files. If a file is linked to that user, remove the link. Then delete the files.

6. Troubleshooting

Most user administration problems can be traced back to ownership and group membership questions or to erroneous entries in the `/etc/passwd` and `/etc/group` files.

The following are error messages and other potential problems, with suggestions for responding.

No home directory

If you see this error message at login time, check that the entry for that user in the `/etc/passwd` file is correct. Typically, if the *home-directory* field is empty but surrounded by colons, the

message at login time is

unable to change directory to ""

No login will occur. If the *home-directory* field is missing (that is, if the *startup-program* field appears directly after the *gid* field), then the message is

unable to change directory to "*program*"

where *program* stands for the entry in the last field of the */etc/passwd* file.

Can't create files '.not found'

Check that the home directory, or the current directory, has the appropriate permissions, ownership, and group membership.

Can't list a directory

The user is in a directory whose group membership is not one of the user's current groups, or the directory's permissions are wrong.

Can't read mail

Check that the file */usr/mail/login-name*, where *login-name* stands for the user's login name, has the appropriate permissions, ownership, and group membership.

Files owned by other user

It is very important to make sure that no two users have the same user ID in the third field of their */etc/passwd* entries. If this happens, some programs get confused.

One such program is *ls*. The information which is stored about a file does not contain the name of the user to which that file belongs; it contains only the user ID. Before *ls* can print the user name when it is invoked with the *-l* flag option, *ls* must go to the */etc/passwd* file to associate that user ID with a login name. Two users with the same user ID unavoidably confuse *ls*.

PLACE TAB
HERE

BACKUPS

BACK OF TAB

Chapter 4

Backing Up Your System

Contents

1. Overview	1
1.1 Backup mediums	2
1.2 Full versus partial backups	2
1.3 A common backup scheme	2
1.4 Referring to devices by device file names	4
1.5 Mounted versus unmounted file systems	6
1.6 The backup utilities	7
1.6.1 <code>cpio</code>	7
1.6.2 <code>tar</code>	8
1.6.3 <code>dump.bsd</code> and <code>restore</code>	9
1.7 The Tape Backup 40SC software	10
2. When to use 3.5-inch disks	11
2.1 Preparing 3.5-inch disks for use	11
2.1.1 Initializing a disk with the Macintosh OS	11
2.1.2 Initializing a disk with A/UX	11
2.1.3 Ejecting disks	12
3. When to use tape	12
3.1 Preparing a tape cartridge for use	13
3.1.1 Formatting a tape cartridge in the Macintosh OS	13
3.1.2 Formatting a tape in A/UX	14
4. Using <code>cpio</code>	15
4.1 <code>cpio</code> and the Tape Backup 40SC	16
4.2 Copying all files in a directory tree to a disk or tape	16
4.3 Creating selective backups	17
4.4 Creating full backups	17

4.5	Creating incremental backups	18
4.6	Listing a table of contents for a disk or tape	19
4.7	Recovering all files on a disk or tape	19
4.8	Recovering selected files from a disk or tape	20
4.9	In the event of hard I/O errors	21
5.	Using tar	21
5.1	When copying to tape	22
5.2	If a backup requires multiple volumes	22
5.3	Copying an entire directory to a disk	23
5.4	Copying specific files	24
5.5	Appending a file to a disk	24
5.6	Adding a later version of a file to a disk or tape	25
5.7	Extracting a specific file	25
5.8	Creating a table of contents from a tar archive	26
5.9	Recovering the latest version of a file	26
5.10	Recovering a particular version of a file	27
6.	The dump , bsd and restore utilities	28
6.1	Dump levels	28
6.1.1	Using dump levels in a monthly backup strategy	29
6.1.2	Restoring from multiple dump levels	29
6.2	Using dump.bsd	30
6.2.1	dump.bsd keys	31
6.3	Using restore	33
6.3.1	Interactive mode for restore	34
6.3.2	restore keys	36
6.3.3	restore options	37
7.	Verifying data on backed-up disks	38
8.	Using the Apple Tape Backup 40SC software	38
8.1	Hints on backing up volumes and partitions	43
8.2	Restoring partitions	43
8.2.1	Hints on restoring volumes and partitions	47

Figures

Figure 4-1. A common backup scheme	3
Figure 4-2. Format Cartridge status screen	14
Figure 4-3. Backup Volumes and Partitions selection screen	40
Figure 4-4. Selecting partitions for back up	41
Figure 4-5. Backup Volumes and Partitions status screen	42
Figure 4-6. Restore Volumes and Partition dialog box	45
Figure 4-7. Selecting a destination for a partition restore	46

BLANK PAGE

Chapter 4

Backing Up Your System

1. Overview

Making regular backup copies of files and file systems is one of the most important duties of the A/UX system administrator. Computer data that is stored on disk can be damaged by hardware failure, or users may accidentally remove it. If you make regular backups, you can restore any data that is damaged, lost, or destroyed.

A backup is data stored on a hard disk copied to an alternate medium, such as a 3.5-inch disk or magnetic tape.

It is a good idea to store backups in a safe place, off-site if necessary. It is also wise to maintain a backup log as a written record of what was backed up, for use in an emergency.

There are many ways to back up data. To decide on the best technique, you should compare the time it takes to complete a backup with the time it may take to restore a backup. You should also consider how often your data is changed, how valuable the data is, and how many people use the system. The safest scheme is to devise an overlapping strategy, combining two or three backup techniques. Generally, if you use a regular schedule for full backups and supplement those with one or two partial backups, you can be assured that you will be able to rebuild your system in the event of a disaster. You may want to customize backup commands in a shell script to keep all backups consistent.

You may also use the A/UX backup utilities to store directories no longer needed on the system. By storing unused data on 3.5-inch disks or tape cartridge, you free the system disk for use and improve performance.

Backing up and restoring are mutually dependent activities, that is, if you backup with `tar`, you can use only `tar` to retrieve the data.

The backup and restore utilities available on the A/UX system include `cpio`, `tar`, and `dump.bsd`. The backup utility under the Macintosh Operating System provides an easy way to make full backups onto a 40 MB tape cartridge.

1.1 Backup mediums

You can make backups on a variety of medium. Currently you can back up A/UX files onto 3.5-inch disks, cartridge tapes (using the Apple Tape Backup 40SC), or nine-track magnetic tape. This chapter explains how to back up onto disks and tape cartridges.

1.2 Full versus partial backups

There are two main strategies for creating backups. The first is a full backup, during which all the data on each file system is copied; it is a time-consuming process. Full backups copy a system in such a way that you can reload it if your disk is completely erased.

The second strategy is a partial backup, during which only part of the system is backed up. This is less time consuming and more useful for most types of everyday work. Depending on which utility you use, you may do either a selective or an incremental partial backup.

To make a selective backup, you specify the files and directories according to your needs, such as specific filenames, or by user or group ownership. Generally, you use `tar` or `cpio` to back up specific data.

To make an incremental backup the system uses the modification dates on files to automatically copy all newly created files and all files modified since the date of the last backup. Although you can use both `tar` and `cpio` to make incremental backups, the `dump.bsd` and `restore` utilities are generally preferred. During incremental backups, the system saves its most recently modified files and routinely backs up files that are changed often, such as `/etc/passwd` and `/etc/groups`. One of the main problems with incremental backups is that it can take a lot of time to locate a file and restore it if you don't know the file's last modification date.

1.3 A common backup scheme

A commonly used A/UX backup scheme is illustrated in Figure 4-1. The figure shows four levels of backup and the appropriate utilities for making those backups.

This scheme ensures that your data is adequately backed up at all times but is not overly cumbersome or time consuming.

Figure 4-1. A common backup scheme

Quarterly backups

Make a full system backup when you first start using the system and again at the end of each quarter. You can either store the quarterly backups off site or create duplicate backup media, storing one set on site and another set off site. Quarterly backup media should be stored for an indefinite period of time. *Be sure to verify that full backups are readable.* See "Verifying Data on Backed-Up Disks," or, if you are using cartridge tape, see "Using the Apple Tape Backup 40SC Software," both in this chapter.

Monthly backups

Make an incremental backup at the end of each month. You should also store these backups in a safe place for 3 to 6 months,

depending on your site's needs and the relative importance of your data. Once you have a monthly backup, you no longer need the previous month's weekly backups, and so you can recycle those media. Be sure to verify that new backups are readable before recycling disks or tapes containing old weekly backups.

Weekly backups

Make an incremental backup at the end of each week to capture all files modified or created during the week. Store these backups in a safe place, and reuse them for the monthly backup at the end of the current month. Once you have a weekly backup, you no longer need the previous week's daily backups, so you can recycle those media. Be sure to verify that new backups are readable before recycling disks or tapes containing old daily backups.

Daily backups

Make an incremental backup at the end of each weekday to copy all newly created files or modified files. Set aside one formatted backup medium (or more if needed) for each day. Then, at the beginning of each new week, copy over the previous week's backup media; on Monday, copy over the previous Monday's backup; on Tuesday, copy over the previous Tuesday's; and so forth.

You can recycle the tapes or disks used for backups once they are no longer useful. Typically a backup is considered useful for two units of time beyond its creation date. For example, Monday's daily backup disk can be recycled Wednesday evening, keeping Tuesday's backup in reserve in case Wednesday's is damaged in some way. Remember that 3.5-inch disks, cartridge tapes, streaming tapes, and certain other backup media can be used only a finite number of times; check the manufacturer's specifications to determine how many times you can safely recycle your backup media.

1.4 Referring to devices by device file names

When you use most backup utilities, you need to specify where the files you want to back up are, and where the backups should be recorded. In A/UX, all peripherals (hard disk drives, 3.5-inch disk drives, tape units, terminals, and so on) are known to the system through device files located in the /dev directory. ("Device" is a synonym for peripheral.) When a backup utility reads or writes to such a file, that

information is read (or written) to the peripheral corresponding to the device file.

There are two varieties of device file: block devices and character devices (also known as "raw" devices). Whether you access the device as a block device or as a raw device depends on the requirements of the utility you are using. The following table lists the names of the device files that are standard for the Macintosh II.

Device File Name	Peripheral	Type
/dev/floppy0	floppy drive #0	block
/dev/rfloppy0	floppy drive #0	char
/dev/floppy1	optional floppy drive #1	block
/dev/rfloppy1	optional floppy drive #1	char
/dev/dsk/c8d0s0	synonym: floppy drive #0	block
/dev/rdsk/c8d0s0	synonym: floppy drive #0	char
/dev/dsk/c0d0s0	built-in hard disk	block
/dev/rdsk/c0d0s0	built-in hard disk	char
/dev/rmt/tc1	tape backup	block

Note: A user (especially the root user) should never attempt to write data directly using the name of a device file that accesses a disk or tape drive. These filenames should be used only on backup utility command lines. Also note that all examples in this chapter use the floppy drive #0, although floppy drive #1 (if this option is present) would work as well.

As shown in Table 4-1, you can use the synonym `/dev/dsk/c8d0s0` for floppy drive #0. This synonym reflects the SCSI naming scheme, in which devices are named in the `/dev/dsk` and `/dev/rdsk` directories according to their controller, drive, and slice number. For example, `/dev/dsk/c8d0s0` signifies SCSI ID 8, the first drive, and the first partition. An external hard disk installed, for example, in slot 5 would be `/dev/dsk/c5d0s0`. See *gd(7)* in *A/UX System Administrator's Reference* for more information on the SCSI naming scheme.

1.5 Mounted versus unmounted file systems

A file system exists on a logical portion of a physical device. This logical portion is called a **partition** and is accessed through the device files explained in the previous section. A file system that is accessible to users is called a **mounted file system**. With the exception of the root file system, which is always mounted, file systems must be explicitly mounted and unmounted with the **mount** and **umount** commands.

Note: Currently the only user-accessible file system on the built-in hard disk is the root file system, which is permanently mounted.

A file system is made accessible by mounting it on a directory of any other mounted file system. This directory can be any ordinary A/UX directory and is called the **mount point** of the file system. When you change your current directory to the mount point, you may traverse the directory tree of this file system as if it were an ordinary branch of the root file system.

If you want to create a file system on a 3.5-inch disk, place a formatted disk in the drive and enter the following command:

```
mkfs /dev/rfloppy0 1600
```

To use this file system, issue the **mkdir** command to create an empty directory on the root file system (we suggest a name such as **/f0**) and then use the **mount** command to mount the file system. The command sequence is

```
mkdir /f0
mount /dev/floppy0 /f0
```

Any files and directories you then create in **/f0** appear as ordinary files and directories, even though they reside on the 3.5-inch disk drive.

To remove the file system disk, enter the commands

```
umount /dev/floppy0
eject
```


Warning: Removing the disk before issuing the `umount` command can damage the file system because A/UX may still have file system data stored in memory.

The error message

`/f0: Device busy`

means that you or other users are accessing files or directories on the file system mounted on `/f0`. If you are in single-user mode, simply change your current directory to a directory that is not on this file system and reenter the `umount` command. If other users are accessing this file system, ask them to finish their work and change their current directory to another so that you can unmount the file system.

1.6 The backup utilities

The A/UX backup utilities fall into two categories, archival and copy utilities. This chapter describes archival utilities, which copy data and reference information pertinent to the data. Archival utilities are designed specifically to move files and directories between media; the reference information is used to reconstruct the original data structure.

The A/UX copy utilities copy data only; they operate on a single file at a time and disregard any structure that that file may have (for instance, the file may be an entire file system). For information on these programs, see `dd(1)` in *A/UX Command Reference* and `volcopy(1M)` in *A/UX System Administrator's Reference*.

The A/UX archival backup utilities include `cpio`, `tar`, and `dump.bsd`. The Tape Backup 40SC software also backs up A/UX partitions and is described in this chapter.

1.6.1 `cpio`

The utility `cpio`, which stands for "copy input to output," is used to back up and restore an entire file system or individual files.

The advantages of using `cpio` include:

- It is a convenient way to copy numerous files that you select by individual name, perhaps because they reside in various directories.
- It is useful for copying a directory and its contents, while leaving out the contents of the subdirectories.

- It not only rebuilds the directory for you when retrieving backed up data but also saves the file permissions, ownerships, and groups.
- It can copy device files, meaning everything stored in `/dev`, whereas `tar` cannot.

The disadvantages of using `cpio` include:

- It is slow, operating at about the same speed as `tar`.
- Its syntax is somewhat irregular.
- It is not convenient for copying the entire contents of a directory tree.
- The backup copies are not transferable between computer systems unless you use the `-c` option.

1.6.2 `tar`

The utility `tar`, which stands for "tape archiver" is used to back up a directory or individual files in a file system. The `tar` command is the best utility to use for transferring files from one system to another, for example, transferring ASCII files between BSD systems and System III- or System V-based systems (such as A/UX). Of course, both systems must have the `tar` utility.

The advantages of using `tar` include:

- You can back up and retrieve individual files, directories, and file systems, or any combination of these three.
- You can reliably transfer files between computer systems, such as between a Macintosh II and a VAX®.
- It accepts a blocking factor (with the `b` option) that lets you match the block size of the output files with the block size of the output device, such as 8K blocks for the Apple Tape Backup 40SC.

The disadvantages of using `tar` include:

- It is the slowest way to copy and retrieve data.
- If you think your backup might not fit on one tape, disk, or other backup medium, you must specify the maximum number of

blocks the medium can hold. For instance, if you do not specify blocks and the copy runs off the end of the tape, you have to start the backup over again.

- It can copy only regular files and not directories or device files.

1.6.3 dump.bsd and restore

The `dump.bsd` utility is used for incremental or full file system backups. The utility supports "dump levels"; these levels range from 0 to 9, where 0 represents a full backup of the entire system and the other numbers are used for incremental backups.

The `restore` program is `dump.bsd`'s companion utility. It retrieves files and directories from a backup medium created with `dump.bsd`. You recover single files from the backup medium by using the `-x` option for the `restore` program (see "Using restore"). When using `restore`, you must be careful not to replace the current file system with an older version of itself.

The advantages of using `dump.bsd` and `restore` include:

- These utilities are reasonably fast. Although `dump.bsd` does not make backups as fast as the Tape Backup 40SC software makes an image dump, it is generally faster than `tar` and `cpio`.
- You can make full backups (of the entire file system) or incremental backups (of specific files).
- They allow you to back up only those files modified or created after a certain date. The utility does this by keeping a record of the last full dump date. Thus, it knows what files to back up when you request an incremental dump.
- They typically require only one 800K disk for a daily backup because of their ability to back up only files that have changed.

The disadvantages of using `dump.bsd` and `restore` include:

- They are more difficult to use than the other backup utilities.
- They operate on file systems. As distributed, A/UX has only one file system, and so this is not a drawback. However, if you add one or more file systems, your use of these backup utilities

becomes that much more complicated because you have to track the file systems individually.

- Backing up a disk that is full or well over 75% full requires more disks or tapes than making an image dump with the Tape Backup 40SC software does. This is due to the overhead information `dump.bsd` puts in an archive.
- You must be sure the system's date and time are always correct, or you are likely to lose files when restoring from an incremental backup.
- Using the `-c` option you can transfer backups made with `tar` and `-cpio` among different computers; that is, a backup made with `tar` on a Macintosh II can be restored with the `tar` command on a VAX. The same is not true for backups made with `dump.bsd`.

1.7 The Tape Backup 40SC software

The advantages of using the Tape Backup 40SC software include:

- It is an easy way to make a full backup because it prompts you through the process with dialog boxes, and because it calculates and lets you know in advance how many tapes and minutes are required to complete the backup.
- It can restore your A/UX disk in the event it fails entirely, because it restores from the Macintosh OS rather than A/UX. This advantage is unique among the backup utilities for A/UX.
- It is faster than `dump.bsd` when making a monthly backup.

The disadvantages of using the Tape Backup 40SC include:

- A partition is the minimum amount of data you can retrieve.
- You need to exit A/UX to use the utility. This may be inconvenient, especially if others want to be using A/UX through attached terminals.
- It makes an image backup, which copies everything on the disk, including blank spaces. As such, it tends to require more tapes than `dump.bsd` does when backing up a disk that is less than half full.

Note: You should generally perform backups (and all major administrative tasks) while the system is running in single-user mode. If you make a backup of a mounted file system that is being altered by frequent writes, you risk backing up an outdated and inconsistent file system.

2. When to use 3.5-inch disks

Use 3.5-inch disks to store backups when:

- backing up a small amount of data, such as several data files
- making a daily backup, that is, backing up files changed that day

2.1 Preparing 3.5-Inch disks for use

Before you can copy files to a 3.5-inch disk, the disk must be initialized; otherwise, you'll see an error message. The initialization process is also called formatting. Formatting can be done in either the Macintosh OS or in A/UX. The Macintosh method has proven more reliable and so is recommended over disk formatting in A/UX.

2.1.1 Initializing a disk with the Macintosh OS

Place the disk into the disk drive (or into the rightmost drive if you have two drives). Since the disk is not initialized, you're informed of this and given three initialization choices: one-sided format, two-sided format, and eject without initializing. In the dialog box, click Two-Sided (A/UX cannot read from single-sided disks). Type a name for the disk, and the disk will be initialized in a few seconds.

If you're initializing a used disk (that is, one containing information) you need to erase the disk first. From the Special menu click Erase Disk. This option erases everything on the disk (after appropriate warnings) and gives you the same choices you have if you are initializing a new disk.

2.1.2 Initializing a disk with A/UX

When initializing a disk with A/UX, be sure the disk contains no valuable information: A/UX initializes without reminding you that all information on the disk will be erased.

To format a 3.5-inch disk, place the disk into the disk drive (or into the rightmost drive if your system has two 3.5-inch drives). Then type

```
diskformat /dev/rfloppy0
```

You will be asked to confirm your request to format the disk; press RETURN to start formatting or type an *interrupt* to abort the operation. When the shell prompt returns, your disk is formatted.

Note: Under normal use, you need to format a disk only once. If A/UX prints messages about read and/or write errors while using backup utilities, you can try formatting the disk again. If the error messages persist, use a new disk. Note also that formatting erases any information that was previously stored on the disk.

You can also instruct A/UX to format a disk when copying data out to a disk with the `cpio` command. You specify formatting with the `-F` option to the `cpio` command. Here is an example:

```
ls | cpio -oF > /dev/rfloppy0
```

You can only use the formatting option to `cpio` when copying to an 800K disk in an 800K disk drive.

2.1.3 Ejecting disks

To eject a disk from the floppy drive 0, by default the rightmost drive, enter

```
eject
```

or enter

```
eject 0
```

To eject a disk from the floppy drive 1, enter

```
eject 1
```

3. When to use tape

A cartridge used by the Apple Tape Backup 40SC holds approximately 38 MB of data, whereas a 3.5-inch disk holds 800K. Therefore, use tape cartridges when you back up large amounts of data, for example, when you make a full backup to safeguard against a system crash. Use tape cartridges when:

- making weekly or monthly backups
- backing up large amounts of data, such as an entire file system

You can use `cpio`, `tar`, `dump.bsd`, and the Tape Backup 40SC software to make backups on cartridge tape.

Complete information on the Apple Tape Backup 40SC, from how to install the unit, prepare cartridges for data, and care for the machine to how to correct problems you might run into in the course of making backups, is provided in *Apple Tape Backup 40SC Owner's Guide*, which is delivered with the unit. Refer to that guide for information not provided in this book.

3.1 Preparing a tape cartridge for use

Formatting a tape cartridge prepares it to receive, store, and transmit data. Apple tape cartridges are preformatted for you, but other brands may not be. You can format tapes in either the Macintosh OS or in A/UX, whichever is convenient.

It is a good idea to format all unformatted tape cartridges as soon as you purchase them. Each tape cartridge stores up to 38.5 MB of data. You'll need more than one tape cartridge to back up a 40 MB hard disk that is completely full.

3.1.1 Formatting a tape cartridge in the Macintosh OS

To format a tape cartridge, follow these steps. The process lasts about 40 minutes and must not be interrupted.

1. From the Macintosh OS, open the Apple Tape Backup 40SC application.
2. Choose Format Cartridge from the pull-down File menu.
 - **Warning:** Never eject the tape cartridge while the red activity light is on or flashing.
3. Follow instructions in the dialog boxes to direct the Format Cartridge operation.

When you insert the tape cartridge, make sure that the record-lock switch is in the unlocked position (to the left).

Push firmly until the tape cartridge is inside the Tape Backup 40SC.

C

C

C

2. Type

`mt format`
and press RETURN.

Formatting requires about 30 minutes. After formatting is complete, the tape is ready to receive data.

4. Using `cpio`

The `cpio` utility copies files to or from the device or location you specify. For example, you can copy files to a disk or to a file system with equal ease.

You cannot give filenames as arguments to `cpio` as you can with `tar`. Instead, `cpio` takes its input from other utilities. The utilities most commonly used to give input to `cpio` are `ls` and `find`. Because `cpio` restores files *relative to the current directory*, *never* use an initial '/' when creating a backup with `cpio`. Instead, move to the directory you want to copy from and then give the `cpio` command.

When using `ls` to provide input to `cpio`, do not use options to `ls` because `cpio` must receive files one per line.

These are the most important options used with `cpio`:

- o Copies *out* files to a device or location you specify, such as disk, tape, or file.
- p Copies (passes) to another directory or file system you specify.
- i Copies *in* files from a device or location you specify, such as disk, tape, or file.

You will notice in later examples that, when `cpio` is used to copy to a 3.5-inch disk, the disk drive is referred to as a raw device rather than a block device. For example, you'll see `cpio -ov > /dev/rfloppy0` rather than `cpio -ov > /dev/floppy0`. The `cpio` utility has been modified to take fuller advantage of the Macintosh disk drive and as a result performs more reliably as a raw device than a block device.

4.1 `cpio` and the Tape Backup 40SC

When you use `cpio` to copy to tape, `cpio` does not know how to recognize the end of the tape. For this reason, the `cpio` command fails when it attempts to copy more files than will fit on one tape (38.5 MB). If the command fails for this reason, you will see the following error message:

```
No such device or address
```

Despite the wording of the error message, you need to reenter the command and specify fewer files to complete the backup successfully.

The tape unit requires that you send the data in 8K blocks. When you use `cpio` to copy to tape, use the `tcb` filter to block the data. The sole purpose of the filter is to block data in 8K blocks.

Here's an example:

```
ls | cpio -ov | tcb > /dev/rmt/tcl
```

The components of this command are as follows:

`ls`

The `ls` command lists the files in the current directory and sends them through a pipe as input to the `cpio` command.

`cpio -ov`

The `cpio` command, with options that copy out files (`o`), display the filenames on the screen (`v`).

`| tcb`

Piping the files through the `tcb` filter, which blocks the data in 8K blocks so the tape unit can read it.

`> /dev/rmt/tcl`

The output device, in this case the tape controller for an Apple Tape Backup 40 SC at (`tc`) SCSI ID 1.

4.2 Copying all files in a directory tree to a disk or tape

You may often need to copy all the files in a directory tree to a 3.5-inch disk or tape. The simplest way to do this is with the following command:

```
find directory name -print | cpio -o > /dev/backupmedium
```

If you are copying to tape, remember to pipe the `cpio` output through the `tcb` filter to block the data in 8K blocks. See the preceding section, "cpio and the Tape Backup 40SC," for further information.

4.3 Creating selective backups

To create selective backups, that is, to back up only those files that fall into a specified category, enter

```
find directory-name -user user-name \  
-print | cpio -ovB > /dev/rfloppy0
```

(This command line has been wrapped onto two lines by placing a backslash at the end of the first line; this is not necessary to the command.)

The components of this command are as follows:

```
find directory-name -user user-name -print
```

The `find` command searches the specified *directory name* to obtain the files of the specified *user-name* and passes the pathnames of the user's files to `cpio`.

```
| cpio -ovB
```

The `cpio` command, with options to copy out files (`o`), display the name of every file copied on the screen (`v`), and block output (`B`) 5120 bytes per record.

```
> /dev/rfloppy0
```

The output medium, in this example the (raw) disk.

You can use other `find` options to select files by other characteristics, such as group ownership or age of the file. See "Creating Incremental Backups," later in this chapter, and `find(1)` in *A/UX Command Reference*.

4.4 Creating full backups

To create a full backup of your entire system, enter

```
find / -print | cpio -ov | tcb > /dev/rmt/tcl
```

The components of this command are as follows:

```
find / -print
```

The `find` command, beginning at the root directory (/), passes the file pathnames to `cpio`.

```
| cpio -ov
```

The `cpio` command, with options that copy out files (`o`) and display the name of every file copied on the screen (`v`).

```
| tcb
```

Piping the files through the `tcb` filter to block the data in 8K blocks.

```
> /dev/rmt/tcl
```

The output medium, in this example the tape controller for an Apple Tape Backup 40 SC at SCSI ID 1.

If the backup will not fit on one tape (38.5 MB), you'll see the error message:

```
No such device or address
```

Despite the wording of the message, you need to reenter the command but specify fewer files to copy, that is, by sending it files from a lower point in the file system.

4.5 Creating Incremental backups

To create incremental backups, that is, to back up only files that have been modified or created since a certain time, enter

```
find / -mtime -1 \  
-print | cpio -ovB > /dev/rfloppy0
```

(This command line has been wrapped onto two lines by placing a backslash at the end of the first line; this is not necessary to the command.)

The components of these commands are as follows:

```
find /
```

The `find` command, beginning at the root directory (/), passes the file pathnames to `cpio`.

-mtime -1 -print

Select files modified (**-mtime**) since the last day (**-1**) and pass the file pathnames to **cpio**. Use **-1** for daily incremental backups and **-7** for weekly incremental backups.

| cpio -ovB

The **cpio** command, with options that copy out files (**o**), print the name every file copied on the screen (**v**), and block output (**B**) 5120 bytes per record.

> /dev/rfloppy0

The output medium, in this example, the disk.

4.6 Listing a table of contents for a disk or tape

To list the table of contents for a 3.5-inch disk or a tape made with **cpio**, enter

cpio -it < /dev/backupmedium

The components of these commands are:

cpio -it

The **cpio** command with the **-i** and **-t** flag options. The **-i** flag option specifies that input filenames should be extracted, and the **-t** option generates a table of contents.

< /dev/backupmedium

Use the files on the backup medium as input to the **cpio** command.

4.7 Recovering all files on a disk or tape

To recover all files from a disk or tape created with **cpio**, enter

cpio -ivdmu < /dev/backupmedium

The components of these commands are as follows:

cpio -ivdmu

The **cpio** command with flag options. The **i** option extracts files from the 3.5-inch disk, the **v** option prints the filename on the screen after it has been extracted, the **d** option creates any directories needed to extract the files, the **m** option preserves the file's modification date, and the **u** option extracts files from the archive unconditionally. (Normally, **cpio** will not extract a file

that is older than an existing file with the same pathname.) See `cpio(1)` in *A/UX Command Reference* for additional information.

`< /dev/backupmedium`

Use the files on the backup medium as input to the `cpio` command.

4.8 Recovering selected files from a disk or tape

To recover only certain files from a disk or tape created with `cpio`, first obtain a list of the full pathnames for files on the disk or tape, using `cpio` with the `-i` and `-t` options:

```
cpio -it < /dev/backupmedium
```

Now select the files you want to extract, and enter

```
cpio -ivdmu filename... < /dev/backupmedium
```

The components of these commands are as follows:

`cpio -ivdmu`

The `cpio` command with flag options. The `-i` option extracts files from the 3.5-inch disk or the tape, the `v` option prints the filename on the screen after it has been extracted, the `-d` option creates any directories needed to extract the files, the `m` option preserves the file's modification date, and the `u` option extracts files from the archive unconditionally. (Normally, `cpio` will not extract a file that is older than an existing file with the same pathname.) See `cpio(1)` in *A/UX Command Reference* for additional information.

filename

The name of the file(s) to be extracted.

Note: You can use file expansion characters such as `*` and `?`, but these characters must be quoted (enclosed in single or double quotes or preceded by a backslash) to prevent the shell from interpreting them before they are passed to `cpio`.

< /dev/backupmedium

Use the files on the backup medium as input to the `cpio` command.

4.9 In the event of hard I/O errors

When you're checking data after you have created backups, you must restart the entire procedure if a hard I/O error occurs while the data is being read. However, before beginning the procedure anew, try reinserting the problem disk or tape; often, the system can read it the second time. If it can't, then you need to start the procedure from the beginning, using a fresh disk or tape.

You cannot interrupt `cpio` to allow formatting for additional disks or tapes. The process must be aborted, fresh disks formatted, and the procedure started again.

5. Using `tar`

The `tar` command copies a single file or groups of files to or from a disk. The files are copied sequentially, and no directory structure is created. Give either full or relative pathnames of the files you want to copy if you will be restoring the files to the same account. If you might be restoring the files to an account other than the one from which you copied them (for example to someone else's account on another computer), then be sure to use relative pathnames. In general, relative pathnames are safer.

You can use `tar` to extract files copied with `tar` from the disk or tape. When retrieving files the files are put into the current directory and maintain the directory structure with which you copied them. For example, if you change to the directory `/user/linda` and copy the file `/user/linda/stories/nickname` by giving the relative pathname `stories/nickname`, when you retrieve this file, it will be copied into the current directory as `stories/nickname`.

The `tar` command gives the capability of adding to files already on the backup media. The `tar` command also displays a table of contents for the files archived on a particular disk or tape.

Options used with `tar` control its actions. Some options are required when copying to the Apple Tape Backup 40SC.

5.1 When copying to tape

By default, `tar` copies data out in 512-byte blocks. This works fine when copying to 3.5-inch disk, because the disks store information in blocks of that size. However, when copying to the Apple Tape Backup 40SC, data must be sent in 8K blocks, and this size must be specified. Here's an example:

```
tar cbf 16 /dev/rmt/tcl usr/lib
```

The components of this command are as follows:

```
tar cbf 16
```

The `tar` command with the option `cbf`. The `c` option creates a new backup, writing at the beginning of the tape. The `b` option alerts `tar` to use the next argument as the block size for sending out the files. The block size for the tape can be any multiple of 512 bytes. Common multiples used are 8K and 16K. In this example, 16 is used because blocks of 16K are copied faster than blocks of 8K, yet 16K blocks are not so large as to cause a large part at the end of the tape to be left blank because a block couldn't fit. The `f` option alerts `tar` to use the next argument, in this case `/dev/rmt/tcl`, as the place in which to copy the files.

```
/dev/rmt/tcl usr/lib
```

The device to copy to, in this case the tape unit with SCSI ID 1, and the files to copy, in this example the directory `usr/lib`.

5.2 If a backup requires multiple volumes

When copying to the Apple Tape Backup 40SC or to a 3.5-inch disk, `tar` does not recognize the end of the medium. It runs past the end of the tape or disk and generates an error message, forcing you to begin copying again. This is understandable since `tar` was designed for use with a nine-track tape, and thus its default length is 2300 feet. To protect against this annoyance, let `tar` know the holding capacity of your backup medium when you suspect your backup might require more than one tape or disk. Adding this argument to your command line is a small price to pay for the assurance of completing your copy successfully.

Here's an example of how to let `tar` know the holding capacity of a cartridge tape:


```
tar cbBf 16 4500 /dev/rmt/tcl usr/lib
```

The `b` option alerts `tar` to use the argument 16 as the blocking factor. The `B` option alerts `tar` that 4500 is the maximum number of blocks the tape can hold.

The exact number of blocks a tape can hold varies, depending on the number of potentially bad blocks on the tape. It is safe to use 4500 blocks as the maximum tape capacity because this figure allows for the maximum number of bad blocks. (This figure also allows for the space required to store the tape's formatting information.) When a tape is formatted, potentially bad blocks are eliminated from the tape's usable space. If you want to know the exact capacity of a formatted tape, use the `mt` command.

Here's an example of using the `mt` command to discover the exact number of usable blocks on a tape:

```
mt -f /dev/rmt/tcl status
```

The command returns a report that includes a line showing the number of available 8K-byte blocks on the tape, as shown in the following:

```
total 4844 blocks (39682048 bytes) avail this cartridge
```

If you are copying to a double-sided, 800K disk, the holding capacity is 1600K. For a single-sided disk, it is 800K.

5.3 Copying an entire directory to a disk

When copying a directory, `tar` copies all its contents, including the contents of its subdirectories.

To copy all files in the `usr/lib` directory, change to the `/` directory and enter the command

```
tar cf /dev/backupmedium usr/lib
```

The components of this command are as follows:

```
tar cf /dev/backupmedium
```

The `tar` command with option `cf`. The `c` option creates a new backup, writing at the beginning of the disk. The `f` option uses the argument `/dev/backupmedium` to archive the files.

`usr/lib`

The relative pathname of the directory to be copied.

Note: If your plan is to place these files into a location other than the one you copied them from, be sure to change to the directory / and then enter `usr/lib`. Otherwise, if you use the absolute pathname, the files will retain that pathname and will not be retrievable from the new location.

5.4 Copying specific files

You will often need to save files that hold important data. Such files include `/etc/passwd` and `/etc/group`. Because these files are crucial to the operation of the system, and because they are frequently modified, they are vulnerable to corruption or even loss.

To copy `/etc/passwd` and `/etc/group`, change to the `/` directory and enter

```
tar cf /dev/backupmedium etc/passwd etc/group
```

The components of this command are as follows:

```
tar cf /dev/backupmedium
```

The `tar` command with option `cf`. The `c` option creates a new backup, writing at the beginning of the disk. The `f` option uses the argument `/dev/backupmedium` to archive the files.

```
etc/passwd etc/group
```

Copy the files `etc/passwd` and `etc/group`.

5.5 Appending a file to a disk

You can append a file or files to an archive already on a 3.5-inch disk (but not to a tape archive). To copy the file `mvusr` from the current directory (`.`) and append it to the files on a disk, enter

```
tar rf /dev/rfloppy0 ./mvusr
```

The components of this command are as follows:

```
tar rf /dev/rfloppy0
```

The `tar` command with options `rf`. The `r` option appends the file to the disk. The `f` option uses the argument `/dev/rfloppy0` as a destination for the files.

`./mvusr`

Write the contents of `mvusr` file from the current directory.

5.6 Adding a later version of a file to a disk or tape

To add a later version of the file `curses.mail` from the current directory (`./`) to a disk or tape, enter

```
tar uvf /dev/backupmedium ./curses.mail
```

The components of this command are as follows:

```
tar uvf /dev/backupmedium
```

The `tar` command with options `uvf`. The `u` option adds the named files to the disk or tape if they are not there or if they are modified. The `v` option displays the file size and filename. The `f` option uses the argument `/dev/backupmedium` to archive the files.

`./curses.mail`

Copy the file `curses.mail` in the current directory.

The `tar` command responds with the message

```
a ./curses.mail 3 blocks
```

If the file has been modified, it is then copied. The `a` indicates that the file has been added to the archive. No message is printed if the file is identical to the copy in the archive.

5.7 Extracting a specific file

To recover a specific file from a 3.5-inch disk created with `tar`, use the following command. In this example, you are recovering `chapter8`.

```
tar xf /dev/backupmedium chapter8
```

The components of this command are as follows:

```
tar xf /dev/backupmedium
```

The `tar` command with option `xf`. The `x` option extracts the specified files from the disk. The `f` option uses the argument `/dev/backupmedium` to archive the files.

`chapter8`

The file to be extracted.

When you use `tar` to extract a file, change directories to the target directory and specify the name as it appears in the `tar` table of contents.

5.8 Creating a table of contents from a `tar` archive

To list the files on a disk or tape created with `tar`, enter

```
tar tvf /dev/backupmedium
```

The components of this command are as follows:

`tar tvf`

The `tar` command with options `tvf`. The `t` option displays a table of contents for the files on the disk. The `v` option displays the file size and filename. The `f` option uses the argument `/dev/backupmedium` to archive the files.

`/dev/backupmedium`

The output medium holding the archive, which could be 3.5-inch disk or tape.

The `tar` command then displays the files with their permissions, ownerships, and dates:

```
rwxr-xr-x102/202    978 Feb  1 14:16 1984 ./envelope
rwxr-xr-x102/202    211 Apr 16 11:29 1984 ./proofread
rwxr-xr-x102/202    978 May 10 10:34 1984 ./envelope
```

The same filename can appear more than once; `tar` allows multiple copies of a file on the same disk.

5.9 Recovering the latest version of a file

To recover the latest version of a file, in this example the file `curses.mail`, enter

```
tar xvf /dev/rfloppy0 ./curses.mail
```

The components of this command are as follows:

`tar xvf`

The `tar` command with options `xvf`. The `x` option extracts the named files. The `v` option displays verbose confirmation that the file was extracted, giving the size and name of the file. The `f` option uses the argument `/dev/rfloppy0` to archive the files.

`/dev/rfloppy0`

The output medium, in this example the disk.

`./curses.mail`

The file to be extracted from the current directory.

The `tar` command then responds with the message

```
x ./curses.mail, 1135 bytes, 3 tape blocks
```

In this example, the `x` at the beginning of the line indicates that the file has been extracted.

5.10 Recovering a particular version of a file

To recover a particular version of a file, determine which version is needed by using the `t` option to display the contents of the disk or tape. See "Creating a Table of Contents from a `tar` Archive," earlier in this chapter.

Once you determine which file you want to extract, use the `w` option with `tar`. When using the `w` option, you must confirm the action you tell the system to take. For example, to extract the May 10 version of `mvusr`, enter

```
tar xvwf /dev/backupmedium ./mvusr
```

The components of this command are as follows:

```
tar xvwf /dev/backupmedium
```

The `tar` command with the options `xvwf`. The `x` option extracts the file from the disk, the `v` option displays the file size and filename, the `w` option causes the system to wait for user confirmation before extracting the file, and the `f` option uses `/dev/backupmedium` as the archive file.

```
./mvusr
```

The file to be extracted in the current directory.

When `tar` displays the correct filename, type `y`, as shown here:

```

x rwxr-xr-x 0/1    140 May  4 18:14 1984 ./mvusr:
x rwxr-xr-x 0/1    162 May 10 10:46 1984 ./mvusr: y
x ./mvusr,
162 bytes, 1 tape blocks
x rwxr-xr-x 0/1    140 May 10 10:48 1984 ./mvusr:

```

A **y** (yes) indicates that the file should be extracted.

Other responses include **n** and **RETURN**, both indicating a "no" response.

6. The `dump.bsd` and `restore` utilities

The `dump.bsd` and `restore` backup utilities are recommended for multi-user installations. When you use `dump.bsd` to create a backup, you must use `restore` to recall a file, directory, or file system.

Note: The `/etc/dumpdates` file must exist or `dump.bsd` prints the error message

```
/etc/dumpdates: No such file or directory
```

Therefore, before you run `dump.bsd` for the first time on your system, create an empty file with the command

```
touch /etc/dumpdates
```

You will not need to enter the above command again unless `/etc/dumpdates` is removed by mistake.

6.1 Dump levels

When you use the `dump.bsd` command, you specify an incremental backup using dump levels, which are integers that can range from 0 through 9. Instead of specifying a date to indicate that you want to back up everything that has been created or modified since that date, you specify a dump level to indicate that you want to back up everything that has been created or modified since you made a backup with a lower dump level.

For example, a level 7 `dump.bsd` backs up all files modified since the most recent backup at dump level 6 or lower. Thus, dump level 0 represents a full backup.

6.1.1 Using dump levels in a monthly backup strategy

Most multi-user installations use a dump strategy based on once-a-month full dumps, as described below:

- Every month do a full dump (level 0):
`dump.bsd 0uF`
- At the end of each week do a weekly dump (level 4):
`dump.bsd 4uF`
- At the end of each working day do a daily dump (level 7):
`dump.bsd 7uF`

The F on the command line tells `dump.bsd` to write the backup to the floppy drive (`/dev/rfloppy0`). By default, `dump.bsd` reads the files to back up from the built-in hard disk (`/dev/rdisk/c0d0s0`). Many keys can alter the default operation of the `dump.bsd` utility. See `dump.bsd(1M)` in *A/UX System Administrator's Reference*.

The level numbers mnemonically represent weeks (4 weeks in a month) and days (7 days in a week). In this strategy, the weekly dump backs up all files modified since the last monthly backup, and daily dumps back up files modified since the last weekly dump.

For the daily and weekly dumps, you can reuse the same backup disks or tapes, overwriting your previous backups. For monthly (level 0) dumps, use a set of fresh disks or tapes and save them for an extended period (generally, 6 months to a year).

6.1.2 Restoring from multiple dump levels

In the following example, the root(/) file system is restored after it was accidentally removed.

To restore the file system to the state in which it existed at the beginning of the month, place the first disk from the current month's level 0 dump disk in the floppy disk drive, and enter the command

```
restore r
```

Note: If the backup archive is comprised of more than one disk (the usual case), `restore` will prompt you when it is ready for you to insert the next disk in the series.

When the level 0 restoration is complete, place the first disk from the current week's level 4 dump disk in the drive and enter

```
restore r
```

If the level 4 backup archive is comprised of more than one disk, `restore` will prompt you when it is ready for you to insert the next disk in the series. When the level 4 restoration is complete, the file system is restored to its state at the beginning of the week.

To complete the restoration, remove the disk from the 3.5-inch drive, place the first disk of yesterday's level 7 dump disk in the drive, and enter

```
restore r
```

When the level 7 restoration is complete, the file system is restored to its state at the time of the most recent backup (last night).

As this example shows, with this scheme for routine backups you need to use only three dump levels to restore an entire file system to its current status.

6.2 Using `dump.bsd`

The `dump.bsd` command operates on the file system mounted on the specified disk partition. It copies all files modified after a certain date to a 3.5-inch disk (or other backup medium).

Because many disks are used to create backups, `dump.bsd` sets a checkpoint for itself at the beginning of each disk. If some error occurs during writing to a disk, `dump.bsd` waits until you have removed the old disk and inserted a new one, then (after prompting with a question) restarts itself from the checkpoint.

The `dump.bsd` utility prompts with questions when any of the following occurs:

- It reaches the end of a disk.
- It reaches the end of a dump.
- A hard I/O error occurs.

You must answer either yes (press `y`) or no (press `n`) to any of `dump.bsd`'s questions.

The following example shows the basic format of a `dump.bsd` command:

```
dump.bsd 0uF
```

This command backs up the file system on `/dev/rdisk/c0d0s0` (the built-in hard disk). The keys `0uF` have these meanings: `0` represents a complete backup (not incremental), `u` updates the system file `/etc/dumpdates` with the time of this backup, and `F` tells `dump.bsd` to write the dump on the 3.5-inch disk drive (`/dev/rfloppy0`).

6.2.1 `dump.bsd` keys

You use keys to control the operations of the `dump.bsd` command. Unless you specify at least one key, `dump.bsd` will not work. Keys are similar to flag options, except that one must be specified. These are the keys:

F Specifies that the dump is to be written to the disk in the 3.5-inch disk drive.

Note: Unless you are writing to an external hard disk or other external device, you should *always* specify this option.

0-9 Specifies the dump level. The `dump.bsd` utility uses this number and the system file `/etc/dumpdates` to determine when the file system was last dumped (at a dump level lower than the number specified) and which files have been modified since.

```
dump.bsd 0F
dump.bsd 4F
```

In these examples, level `0` represents a full backup, whereas level `4` backs up only those files modified since a level `3` or lower-level backup.

- u Writes the date of the beginning of the dump to the file `/etc/dumpdates`. The file records a separate date for each file system and each dump level.

```
dump.bsd 7uF
```

Note that `/etc/dumpdates` must exist or `dump.bsd` will print an error message. Therefore, before you run `dump.bsd` for the first time on your system, create an empty file with the command

```
touch /etc/dumpdates
```

You will not need to enter the above command again unless `/etc/dumpdates` is removed by mistake.

f filename

Backs up data to the specified device or file, other than the default disk. If the filename is `-`, `dump.bsd` writes to the standard output, in which case it can be used as part of a pipeline.

```
dump.bsd 4uf /dev/rdisk/c5d0s0 /dev/rdisk/c0d0s0
```

In this example, the contents of `/dev/rdisk/c0d0s0` (the root file system on the internal hard disk) are written to `/dev/rdisk/c5d0s0` (in this case, an external disk).

Note: Be very careful not to transpose the name of file being written (the first filename argument) and the name of file being read (the second filename argument). An empty file system is the likely result of such an action.

- w Displays the file system that needs to be dumped. The information is gathered from the files `/etc/dumpdates` and `/etc/mtab`.

When using `w`, `dump.bsd` displays the most recent dump date and level for each file system in `/etc/dumpdates`. The file systems that need to be dumped are highlighted.

All other options are ignored when `w` is used, and `dump.bsd` exits immediately.

`dump.bsd W`

Last dump(s) done (Dump '>' file systems):

> /dev/rdisk/c0d0s0 (/)

Last dump: Level 5,

Date Sun Nov 23 16:21

/dev/rdisk/c0d0s2 (/usr)

Last dump: Level 0,

Date Sun Nov 30 22:01

In this example, the file system to be dumped is preceded by the
> symbol and not highlighted.

- w Similar to W, but displays only those file systems that need to be
dumped.

`dump.bsd w`

Dump these file systems:

/dev/rdisk/c0d0s0 (/)

Last dump: Level 5,

Date Sun Nov 23 16:21

- n Notifies all operators in the group operator that `dump.bsd`
requires attention.

`dump.bsd 0un /dev/dsk/c0d0s0`

DUMP: NEEDS ATTENTION: Do you want to
abort dump?:

("yes" or "no") yes

DUMP: The ENTIRE dump is aborted.

6.3 Using `restore`

The `restore` command reads the backup media created with the
`dump.bsd` command. The following example illustrates the basic
form of the `restore` command:

`restore r`

This command reads the default disk drive (/dev/rfloppy0) and
expects to find a disk containing a previously recorded `dump.bsd`

archive. If the archive spans multiple disks (the usual case), `restore` expects to read the first disk of the archive.

The `r` key tells `restore` to load the entire contents of the archive into the current directory. (The `restore` command will recreate the entire file and directory hierarchy of the archive beginning with the current directory.)

Note: The `r` key should be used only to restore a complete `dump.bsd` archive onto an empty hierarchy or to restore an incremental `dump.bsd` archive after a full level 0 restore. Be very careful about where you are in the file system when you use the `r` key. If you start `restore r` from the top of a full hierarchy, you will replace current files with older versions.

Like the `dump.bsd` command, the `restore` command requires keys to control its actions and accepts other arguments to specify files or directories to be restored. See "restore Keys" for more information.

6.3.1 Interactive mode for `restore`

The `restore` command features an interactive mode for extracting files from a dumped disk. You can use `i` with `restore`, as in:

```
restore i
```

When you use `restore` in the interactive mode, it reads the directory information from the disk and then creates a shell-like interface complete with the following prompt:

```
restore >
```

This interface lets you move around the directory tree, selecting files to be extracted. The interface also supports commands that aid in locating files.

The commands are described here. If a command needs an argument and one is not provided, the current directory is used by default.

ls [*arg*]

Displays the contents of the current directory or the specified directory used as its argument. In the display output,

- Directory names are appended with a slash (/).
- Entries selected for extraction are prefixed with an asterisk (*).
- If the `v` key (verbose) is set, the inode number for each entry is also displayed.

`cd arg`

Changes the current working directory to the directory specified as its argument.

`pwd`

Displays the full pathname for the current working directory.

`add [arg]`

Adds the current directory or specified directory to the list of files to be extracted. If a directory is used as an argument, it is recursively extracted, unless the `h` option is used in the `restore` command line.

`delete [arg]`

Deletes the current directory or specified directory from the list of files to be extracted. If a directory is used as an argument, it is recursively extracted, unless the `h` option is used in the `restore` command line.

The easiest way to extract most files from a directory is to add the directory to the extraction list and then delete those files not needed.

`extract`

Extracts all the files on the extraction list from the dumped disk. Then, `restore` asks which volume you want to mount. The quickest way to extract a few files is to start with the last disk and work toward the first.

`setmodes`

In all directories added to the extraction list, the owner, modes, and times are set. If a `restore` is prematurely aborted, this option preserves directories' ownership, modes, and times. Nothing is extracted from the backup medium.

`verbose`

The `ls` command displays the inode number of all entries. The

restore command also displays information for each file extracted.

help

Displays a list of all available commands in the interactive mode.

quit

Immediately terminates the restore program, even if all files or directories are not extracted.

6.3.2 restore keys

These are the keys most commonly used with the restore command. Examples illustrate how to use the keys.

- r Reads and loads the contents of the disk to the current directory. This key should be used only to restore a complete dump tape onto a clear file system, or to restore an incremental dump tape after a full level 0 dump.

Note: Be very careful about where you are in the file system when you use the r key. If you start restore r from the top of a full hierarchy, you will replace current files with older versions.

- R Used when a restore has been interrupted. It requests a particular disk from a multivolume disk set to restart a full restoration.

restore R

x [*arg*]

Specifies files to be extracted from the disk.

If no file or directory is specified, restore begins recursively extracting from the root directory; the entire file system is extracted. If a directory name is specified, it also is recursively extracted, unless the h option is used.

The quickest way to extract a few files is to begin with the last disk and work toward the first. For example, if you use the command

restore x *filename*

only the file represented by *filename* is extracted from the backup medium. The command

`restore xh directory-name`

extracts files from the directory represented by *directory-name*.

The command

`restore x directory-name`

recursively extracts the entire directory hierarchy represented by *directory-name*.

t [*arg*]

Lists the contents of the backup medium. If a filename or directory name is used as an argument, the corresponding files that reside on the disk are listed. As with the *x* option, a directory is recursively listed, unless the *h* option is also used.

6.3.3 restore options

The `restore` command can also be used with options that accompany keys. These are the options, along with examples illustrating how to use them.

f *filename*

Counterpart of the *f* option to the `dump` command. The *f* option used with a filename argument restores data from the file instead of the disk. If `-` is used as the filename, `restore` restores from the standard input, allowing `restore` to be used as part of a pipeline.

`restore rf /tmp/save.level4`

v Stands for "verbose." During a restore, the filename and file type are displayed on the standard output.

`restore rv`

y Causes `restore` to display a prompt asking whether to abort the restoration if a hard I/O error occurs. Otherwise, `restore` tries to skip a bad disk block and continues.

`restore ry`

In the event of a hard I/O error, the `restore` command responds with the message

Should I abort restore? yes or no

- h Extracts the actual directory and not the files that it references.
This prevents hierarchical restoration of complete subtrees from the disk. You restore complete subtrees with the `x` key, described in the previous section, "restore Keys."

`restore rh directory-name`

7. Verifying data on backed-up disks

The `dd` command is used for all backup methods to find hard I/O errors. After that, the appropriate option for each backup method is used to display a table of contents.

For example, for 3.5-inch disks, insert each disk and enter

```
dd if=/dev/rfloppy0 of=/dev/null bs=90b
```

The components of this command are as follows:

<code>dd</code>	command name
<code>if=</code>	input filename
<code>/dev/rfloppy0</code>	input file (floppy disk)
<code>of=</code>	output filename
<code>/dev/null</code>	output file (special file used to discard output)
<code>bs=90b</code>	sets both input and output block size to 90 blocks

If the data is successfully read, messages like the following appear:

```
17+1 blocks in
17+1 blocks out
```

If messages appear indicating hard I/O errors on any of the disks, you need to restart the entire backup, using newly formatted disks to replace the faulty ones.

8. Using the Apple Tape Backup 40SC software

You can use any of the A/UX backup utilities—`tar`, `cpio`, and `dump.bsd`—to make backups on the Apple Tape Backup 40SC tape unit. You can also use the Apple Tape Backup 40SC software. The Apple Tape Backup 40SC tape unit and software are explained in detail

in *Apple Tape Backup 40SC Owner's Guide*. For your convenience, the information from that book on backing up A/UX partitions is provided here.

You can use the Backup/Restore menu of the Apple Tape Backup 40SC to make a full or "image" backup. Everything in the partition that you select is copied, even blank spaces on the disk. You cannot use this utility to back up anything smaller than a partition; the Backup Files and Restore Files commands on the menu are available to back up only Macintosh files, not A/UX files.

Each of the Backup/Restore commands prompts you with dialog boxes. A dialog box also alerts you if you are about to do something that could cause you to lose information on the tape and gives you a chance either to cancel what you were about to do or to switch cartridges.

The Backup Volumes and Partitions operation copies partitions to a tape cartridge, or to several cartridges if the partition exceeds 38.5 MB. Partitions are divisions of a disk for use with different operating systems or file systems. They are explained in Chapter 5, "Preparing an Apple HD SC for A/UX." A disk may have one or more partitions. You can select any combination of partitions to back up. The application copies all information from each selected partition onto a tape cartridge in one process. You can use the tape cartridge copy to restore the data at any time.

Follow these steps to back up a partition:

1. Be sure to have enough formatted tape cartridges on hand to complete the backup. For information on formatting a tape, see "Preparing a Tape Cartridge for Use," earlier in this chapter.
2. Shut down A/UX and reboot the system from the Tape Backup 40SC 3.5-inch disk.
3. Open the Apple Tape Backup 40SC application.
4. Choose the Backup Volumes and Partitions command from the Backup/Restore menu.



Backup Volumes and Partitions operation, the application checks whether or not the tape cartridge contains data. If the tape cartridge contains information, the application displays a warning and allows you either to switch the tape cartridge or to cancel the operation.

You can cancel the Backup Volumes and Partitions operation in midstream, if necessary. All the volumes and partitions copied before you canceled the operation remain copied onto the tape cartridge and can be restored on the hard disk.

8.1 Hints on backing up volumes and partitions

Plan to start the backup operation when you won't need the computer for at least 30 minutes. You can set up the computer to carry out the backup operation while you eat lunch, or at night after you go home (as long as you don't need to insert more tape cartridges).

If your partition contains more than 38.5 MB of data, you'll need more than one cartridge.

After you copy information from your hard disk onto a tape cartridge, you may want to slide the record-lock switch to the locked position to guard against copying over information by accident.

8.2 Restoring partitions

The Restore Volumes and Partitions operation writes all the data from a tape cartridge copy of a partition to your hard disk. The partition you are restoring from tape should be the same size as the disk partition into which you are restoring.

Note: If the partition you are restoring from tape is smaller than the disk partition into which you are restoring, you stand to lose disk space because the disk partition takes on the size of the partition on tape. The maximum disk space you can lose in such a situation is 4 MB. The application will warn you of the potential loss of disk space and give you a chance to cancel the operation. You can then repartition the disk before restoring the tape cartridge copy. You need to cancel the restore

operation and create a partition of the correct size by using HD SC Setup. See Chapter 5, "Preparing an Apple HD SC for A/UX," for further instructions on creating partitions.

Follow these steps to restore a volume.

If your hard disk suffered damage so severe that you have had to re-initialize the disk, use HD SC Setup version 2.0 or later to repartition your disk. (HD SC Setup version 2.0 is included on the Tape Backup 40SC 3.5-inch disks and described in Chapter 5 of this manual.)

1. Shut down A/UX.
2. Reboot from the Tape Backup 40SC 3.5-inch disk.

When planning to restore a partition, you always need to start up from the Tape Backup 40SC 3.5-inch disk. The Tape Backup 40SC program won't allow you to restore over the system file or the application file in use on your hard disk. For this reason, the Tape Backup 40SC 3.5-inch disk contains a system folder so that you can use it as a startup disk.

3. Open the Apple Tape Backup 40SC application from the 3.5-inch disk icon.
4. Insert the first tape cartridge of your backup into the tape drive.
5. Choose the Restore Volumes and Partitions command from the Backup/Restore menu.

A dialog box like the one in Figure 4-6 appears, showing you the volumes and partitions available for the Restore Volumes and Partitions operation. You see a list of destinations to which the copied information can be restored. For a disk with multiple partitions, the dialog box displays each partition as a separate choice. The screen identifies each source partition (the tape cartridge copy you made by using the Backup Volumes and Partitions operation) by displaying its partition name and partition size.

8. Click Restore to begin the restore operation.
9. Follow the instructions in the dialog boxes to direct the volume restore operation.
10. When the restore operation is complete, click OK in the completion message to end the volume restore operation.
11. Eject the tape cartridge.
12. Restart A/UX.

You can cancel the Restore Volumes and Partitions operation in midstream, if necessary. When you cancel the Restore Volumes and Partitions operation, the partition being restored on the hard disk becomes useless. (Partitions already restored during this restoration session are unaffected by the cancel.) Canceling the Restore Volumes and Partitions operation does not alter the tape cartridge copy. You can still restore it to your hard disk. You may have to re-initialize the hard disk and try the Restore Volumes and Partitions command again.

8.2.1 Hints on restoring volumes and partitions

Use the restore operation to reconstruct your hard disk if you have a catastrophe that ruins the data on your hard disk (and your day). The Restore Volumes and Partitions operation writes the contents of a hard disk from a tape cartridge back onto a hard disk.

If your hard disk crashes, you may need to re-initialize the hard disk and then start up from the Tape Backup 40SC 3.5-inch disk to perform the restore operation. See "When Reconfiguring Partitions" in Chapter 5, or your hard disk owner's guide for more information.

During the restore operation, the Tape Backup 40SC reads from the tape cartridge and then writes data onto the hard disk. The restore operation writes over any data that is currently on the hard disk. Dialog boxes warn you of the potential data loss and give you a chance to cancel the operation.

BLANK PAGE

PLACE TAB
HERE

PREPARING A
HARD DISK

BACK OF TAB

Chapter 5

Preparing an Apple HD SC for A/UX

Contents

1. What this chapter includes	1
1.1 Why disk subdivisions are beneficial	2
1.2 Benefits of using the HD SC Setup	2
1.3 Ensuring Apple HD SC compatibility with A/UX	3
1.4 Hardware considerations before you begin	4
2. Background on HD SC Setup	5
3. Background on A/UX file systems	7
3.1 The three steps of a file access	9
3.1.1 Making partitions A/UX-specific: slice numbers	10
3.1.2 A user's perception of slice numbers	11
3.1.3 The administrator's role with slice numbers	14
3.2 The methods of choosing a partition	16
3.3 Using partition administration commands	17
4. The general steps in creating A/UX file systems	18
4.1 What <code>mkfs</code> writes onto a partition	18
4.2 When reconfiguring partitions	19
4.2.1 When re-initializing an error-prone disk	19
5. How to create an A/UX file system of maximum size	20
6. About creating custom partitions	32
6.1 How to create a custom partition	33
7. Subprocedures for creating custom partitions	34
7.1 To remove a partition	34

7.2 To add a partition	35
7.3 To group partitions	37
7.4 To move a partition	38
7.5 To view your partitions	38
8. Quitting HD SC Setup	40
9. Making an A/UX file system partition	40
9.1 Making and mounting an A/UX file system	40
9.2 Using <code>dp</code> and <code>pname</code>	46
10. Adding swap space	51

Figures

Figure 5-1. The main Apple HD SC Setup dialog box	4
Figure 5-2. Creating disk partitions	7
Figure 5-3. Logical file systems versus hardware	9
Figure 5-4. File access sequence for A/UX	10
Figure 5-5. The mounting of file systems	13
Figure 5-6. Relating a file to a mount point	14
Figure 5-7. SCSI ID selector switch and indicator	21
Figure 5-8. The Partition dialog box	23
Figure 5-9. The Custom Partition dialog box	24
Figure 5-10. Adding a partition	26
Figure 5-11. The Partition Type dialog box	28
Figure 5-12. The Details window with a highlighted partition	39
Figure 5-13. Interpreting the partition information from <code>pname</code>	52

Tables

Table 5-1. A/UX partition types available	36
--	-----------

BLANK PAGE

Chapter 5

Preparing an Apple HD SC for A/UX

1. What this chapter includes

This chapter describes how to prepare an Apple hard disk (HD) SC to receive an A/UX file system. (If you have a non-Apple hard disk, see the manual accompanying the disk for instructions on preparing it for use.) Preparing a hard disk simply means partitioning the disk to hold distinct types of data. Partitioning a disk is a formal way to prepare the disk to store and retrieve similar types of data and files from the same place on the disk. In nearly all cases, partitioning an SC hard disk is done with the Apple HD SC Setup program. The SC in the name comes from the interface that connects hard disks to Apple computers; the interface is called SCSI (Small Computer System Interface).

There are a number of ways to partition a disk for A/UX, depending on what you want to put on the disk. This chapter describes all the various ways to partition an Apple HD SC, including the easiest way to prepare one for A/UX.

This chapter describes how to:

- ensure compatibility between Apple hard disk SCs, and between the disks and A/UX (see "Ensuring Apple HD SC Compatibility with A/UX")
- partition an auxiliary disk to contain only one A/UX partition of maximum size (see "How to Create an A/UX File System of Maximum Size")
- partition an auxiliary disk to contain multiple partitions (see "How to Create Custom Partitions")
- recover from disk data errors by using HD SC Setup (see "When Reconfiguring Partitions")
- change the partitioning scheme of an Apple HD SC that already contains data (see "When Reconfiguring Partitions")

- use an auxiliary hard disk partition for additional swap space (see "Adding Swap Space")
- use `dp` and `pname` as necessary to make A/UX recognize certain partitions created using HD SC Setup (see "Using `dp` and `pname`")

1.1 Why disk subdivisions are beneficial

With HD SC Setup 2.0, you can subdivide a hard disk into logical sections, called **partitions**. Partitions allow a disk to accommodate multiple file systems and even multiple operating systems. For example, the A/UX distribution disk contains a partition for the A/UX root file system and a partition for the Macintosh Operating System.

Storing data in separate partitions saves you time and memory space when you make backups. By putting system files in one partition and data files in another, you can easily manage them separately. When making backups, you can concentrate on backing up the partition holding the data files, which change often and thus need backing up frequently. You can ignore the partition that holds the system files, which seldom change and thus seldom need backing up. In general, backups can often be administered more efficiently when files are thoughtfully distributed among disk partitions.

Multiple A/UX partitions can exist on one hard disk or on several hard disks if you have them. If you have an additional hard disk, you can place your data files in a separate partition on the auxiliary disk. It is not advisable to repartition your distribution disk.

1.2 Benefits of using the HD SC Setup

The benefits of the HD SC Setup include the following:

- It allows you to create an unlimited variety of partitioning schemes to subdivide your disk.
- It enables the use of the Tape Backup 40 SC software to back up and restore A/UX partitions.
- It performs all of the arithmetic tasks necessary to arrive at the correct starting disk block for each partition, making it easier to use and more reliable than the corresponding A/UX programs.

- It alters the setup information recorded on the drive to ensure compatibility between A/UX and all levels of Apple HD SC hardware currently available.

Although you can perform all partitioning and initialization functions from within A/UX, only the use of this version of HD SC Setup will ensure the correct operation of A/UX with all Apple HD SC hardware.

1.3 Ensuring Apple HD SC compatibility with A/UX

The following steps describe how to use HD SC Setup to ensure compatibility between A/UX and all levels of Apple HD SC hardware.

Note that you need to perform this procedure only when you do not plan to follow any of the other procedures in this chapter. For example, suppose you have stored data on an Apple hard disk and you do not want to change the partitioning of the disk. In this case, you should still complete this short procedure to eliminate the possibility that A/UX is incompatible with the disk.

1. Start or restart your computer after inserting the Apple HD SC Setup disk in the 3.5-inch disk drive.
2. Open the HD SC Setup application by double-clicking its icon, or by clicking its icon to highlight it, and then choosing Open from the File menu.

A dialog box appears, as shown in Figure 5-1.

the Macintosh OS. You may also want to refer to Chapter 4, "Bridging A/UX and the Macintosh Operating System" in *Getting Started with A/UX*. There you will find instructions on switching back and forth between the two operating systems.

For disks other than the Apple hard disk SCs, refer to the manufacturer's instructions. Note that HD SC Setup 2.0 is intended for use with Apple HD SCs only.

2. Background on HD SC Setup

The HD SC Setup program is primarily used to initialize and partition disks. It was created for use with the Macintosh OS. When it is used with A/UX, many more partitioning possibilities are available.

HD SC Setup creates a partition by storing specific information in a specially reserved area on the disk called a **partition map**. To be accessible, each partition on a disk must have an entry in the disk partition map (DPM). A DPM entry includes the starting block, length, and name of each partition on the disk.

If the HD SC Setup program cannot find a valid disk partition map at the outset, it will not allow you to perform partitioning operations — only the Initialize option will be enabled for use at the opening dialog.

The HD SC Setup program offers you a variety of functions from the opening dialog, including disk initialization and disk partitioning. When you choose disk initialization, the system tests the disk by writing various data patterns to all locations on the disk and verifying them. *This initialization erases all the information previously on the disk.* Additionally, the system builds a bad block table, flagging any defective areas found on the disk so that they won't be used.

After that, HD SC Setup builds a tentative partition map, indicating that one Macintosh partition of maximum size is desired.

Next, HD SC Setup asks you to provide a name for the Macintosh partition it has automatically created. It does this despite the fact that many A/UX users will not want such a partition. If you do not select the partitioning function at this point and instead click Quit, this tentative partition map is written in the partition map on the disk. Figure 5-2 represents this process with arrows leading from HD SC Setup to the partition map fields labeled Name1, Name2, Type1,

Type2, and so on (the partition map actually includes many more fields than those shown, as described in `dpme(4)`). As shown in Figure 5-2, the offset stored in the partition map corresponds to the absolute starting disk block: Offset1 is 4000, and the A/UX root partition is likewise shown to start at that location.

Note that, during partitioning, you do not access the partitions themselves. You only need alter the partition map, and this in turn changes the configuration of partitions.

Another way to alter the partition map is with the `dp` utility. Use this method only to change some noncritical component of the partition map, such as a partition name. Otherwise, using `dp` can easily disrupt the partition map, for instance, by creating overlapping, unreliable partitions. See "Using `dp` and `pname`" for information about using `dp` to make small changes to a partition map.

When you alter the partition map by resizing or rearranging any of the partitions, the Macintosh and A/UX operating systems lose their ability to locate the start of the old partitions and any files within them. In effect, you lose all the old files. This will become clearer when you learn more about the details of an A/UX disk access in the following section.

worthwhile to partition a given disk with more than one Macintosh partition, although HD SC Setup lets you do so.

Only the administrator responsible for maintaining these configurations must know about the special A/UX file system administration commands. Other users can interact with the file systems transparently and need not know what hardware or logical partitions the operating system is accessing during a standard file access. This feature of A/UX is illustrated in Figure 5-3. As the figure shows, file systems A and B could reside together on one disk or separately on two disks.

A/UX supports access to files as long as they are part of an A/UX file system. A file system is a regular data structure that can contain files and directories. A file system is just one type of structure that may be placed inside of a partition. For an A/UX file system to be useful, the beginning of the file system must coincide with the beginning of a partition. Then, the starting offset block of the partition can be interpreted as the first block of the file system superblock. This process is illustrated in Figure 5-4 as step 2 of the file access sequence.

been identified for A/UX use. A/UX locates the partitions that contain valid A/UX file systems by first reading the partition map.

A/UX slice numbers allow for multiple, simultaneously active, and user-configurable file systems that can be accessed through a much simpler user interface.

A partition that has been identified for A/UX use is assigned a slice number. The slice numbers are A/UX-specific: A/UX performs the service of translating slice numbers into the associated partition locations on the disk. Sometimes you must use special A/UX programs to force A/UX to associate a slice number with an otherwise unrecognized partition.

Many UNIX® systems employ the term *slice* in place of *partition*. It is not entirely correct to say that A/UX employs slice numbers to distinguish partitions. Partitions always exist. Slice numbers exist for a partition only if the partition has been recognized for A/UX use (slice numbers are never associated with Macintosh partitions). Because there is no one-to-one correspondence between a slice and a partition, calling a partition a slice can create confusion. A partition that has an associated slice number is more meaningful because it brings the partition into the context of UNIX.

A slice number alone is insufficient to reference a partition. For this reason, the SCSI ID number is also used, and both are made a part of a filename construct that really represents a logical disk device, as described in the section "Using Partition Administration Commands." Only selected administrative commands require you to reference a partition by slice numbers, since this kind of access is profoundly different from the kind that is moderated through a mount point.

When a partition contains an A/UX file system, you can use `mount(1M)` to enable normal (simplified) access to the file system. However, a slice number must be associated with the partition first.

3.1.2 A user's perception of slice numbers

The user's view of a file system is highly regular, since most of the exceptional details about it are disguised to appear as something more familiar. This is a common occurrence within A/UX. For example, a terminal can be accessed from what appears to be a filename. A filename such as this is one of those "exceptional" objects (a terminal)

that is represented as a more familiar object (a file). It not only appears as the more familiar object but also can be acted upon with many of the same commands that apply to the more familiar object.

Likewise, the directory that is the start point for another file system is treated like any other directory. A directory is analogous to a Macintosh folder, a structure that can contain other nested directories and files.

From the user's point of view, an A/UX file system is a collection of directories and files that are all subordinate to a directory that is used in a special way, called a mount point. By just looking at it, you cannot tell that a directory is a mount point. When a mount point appears in a listing (see `ls(1)`), it seems like any other directory.

Figure 5-5 illustrates the use of mount point, in this case `/groups` and `/users`. The curved lines with arrowheads represent the mounting process. Before file systems are mounted on them, the `/groups` and `/users` directories would typically be empty.

great regularity, it soon becomes the name identifying the file system.

As a practical matter, the mount point is what the user needs to know to gain access to the files in the file system. Even though the mount point is a symbolic name for the file system, it is the name that the user must remember. Certainly, remembering the name is easier than remembering the slice number. However, the system administrator can't completely forget the slice number.

As the system administrator, you control the symbolic name for a file system simply by using the normal file renaming command, `mv(1)`, but this works only while the file system is unmounted.

But, the symbolic way of referencing a file system is available only after you mount the file system by using the `mount(1M)` command. Before that, you must use alternate ways of referencing the file system. Without a doubt, these alternate methods require greater understanding of partitions, slice numbers, file systems, and mount points. You need to know how all these provisions interrelate.

The mounting process is the platform that bridges the partition-referencing slice numbers and the file-system referencing mount point. The existence of a mount table entry for the mount point acknowledges that the starting block offset to a partition is known and that it should also be interpreted as the first block in a valid A/UX file system. The association of a slice number with a partition acknowledges that the starting block of a partition is known and that it is available for A/UX use. Then certain "partition-level" or "file-system-level" operations can be performed upon the associated partition (see "Using Partition Administration Commands").

It is not hard to imagine some error conditions that would foul up operations considerably. If A/UX has no business poking around in a Macintosh partition, then such a partition should never be allowed to become associated with an A/UX slice number. Similarly, a partition that does not contain an A/UX file system (such as swap space) should never be allowed to become associated with a mount point. The programs that govern these resources (`mkfs`, `mount`, `umount`, and `pname`) must work to avoid invalid associations such as these.

3.2 The methods of choosing a partition

The remainder of this chapter makes extensive reference to the A/UX device filenames used to identify the HD SCs and the partitions within them. This is because the commands available to manipulate file systems (`mkfs`, `mount`, and others) must be able to access file systems without going through the mount table.

For the administration of disks within A/UX, you must use the slice number and the SCSI ID number to reference a partition on a particular disk. Since these are placed in what appears to be just a normal A/UX pathname, you will need to know the format of that pathname as described in "Using Partition Administration Commands."

The good news is that A/UX tries to correlate partitions to slice numbers according to some simple rules. A/UX automatically gives fixed slice numbers to partitions created within HD SC Setup, as long as they are selected to be certain A/UX types of partitions (as shown in Table 5-1, later in this chapter). However, if a partition is not yet associated with a slice number, you need to use `pname` or `dp` to create this association. Until a slice number identifies the partition, you cannot perform any partition administration tasks, such as creating or mounting a file system partition. Of course, if there is no slice number, you must use `pname` and `dp` to reference partitions.

Within HD SC Setup, referencing a partition is not much of an issue for several reasons. The user need only click a screen graphic that represents all of the partitions available. From a system viewpoint, HD SC Setup does not actually have to make disk accesses into the partitions to create them, since the program merely updates the partition map at a known disk location.

Because HD SC Setup and A/UX use different partition referencing methods, you need to know how they relate to correlate the partitions of one to those of the other.

Thus, you may sometimes be forced to use the `dp` and `pname` utilities to make A/UX recognize a partition by its slice number. The difficulty of doing this is compounded because these A/UX utilities use the partition name and the partition index number to reference partitions. Unfortunately, HD SC Setup does not automatically show you the index number or the partition name to help you correlate what you see

in HD SC Setup with what you see when you use `dp` and `pname`. However, HD SC Setup does show you the partition type, which is very similar to the partition name—enough so that the partition name can be of great help when you must use `dp` and `pname` to force A/UX to recognize a partition (a procedure for this is included in “Using `dp` and `pname`”).

3.3 Using partition administration commands

Most of the commands you use to administer partitions require you to specify a partition. By constructing the appropriate pathname, you specify which disk partition is the object of the administrative operation (such as `mkfs`, `mount`, or `fsck`). Certain components of this pathname vary, since the SCSI device number and a slice number are parts of the pathname.

For most administrative purposes, you reference a particular partition on particular HD SCs according to the following pathname format:

`/dev/dsk/cndmsy`

The directory `/dev` lists all of the devices under A/UX. The subdirectory `dsk` lists the block devices that you can mount as A/UX file systems, for instance, the HD SCs.

The value of *n* after *c* is the SCSI ID of the HD SC; the value of *m* after *d* is the number of the subdrive at that SCSI ID; and the value *y* after *s* is the slice number associated with a particular disk partition.

Some controller boards support multiple disk drives from one SCSI ID, and *m* selects that second drive. However, all of the HD SCs that you connect to the Macintosh II through its built-in SCSI port have separate SCSI ID numbers; none will be identified as subdrives. For example, the internal hard disk on a Macintosh II is identified by the device name `/dev/dsk/c0d0sy`; the first external HD SC—when given SCSI ID number 5, as is the convention—is identified as `/dev/dsk/c5d0sy`. The following is a short list of conventions that Apple currently employs:

- Slice 31 always refers to the entire disk.

- For the boot drive, slice 0 always refers to the root partition.
- For the boot drive, slice 1 is used for the swap partition.

4. The general steps in creating A/UX file systems

The complete procedure for creating custom-size A/UX file systems involves two main steps:

1. Boot the Macintosh operating system and use HD SC Setup 2.0 to allocate A/UX partitions.
2. Boot A/UX and use `mkfs` to create a file system for each A/UX partition in which you want an A/UX file system.

4.1 What `mkfs` writes onto a partition

The `mkfs` program creates file systems by placing the correct initial values into a superblock and storing them at the starting block offset for the associated partition.

To run `mkfs`, you must supply a slice number and SCSI ID numbers as part of a special filename construct. This construct identifies the partition and disk to receive the file system (see "Using Partition Administration Commands," earlier in this chapter).

The file system initially created by `mkfs` has no files. All the blocks available for data storage are placed on the free list.

One of the optional parameters you can supply when you run `mkfs` is the number of inodes you wish to create.

An inode is another data structure nested inside the superblock, which includes a list of pointers to blocks. Some of these pointers are used to point directly to the first few blocks allocated to the associated file. The remaining pointers are known as indirect pointers; they point to supplemental blocks, the contents of which are interpreted as additional direct or indirect pointers.

By specifying the number of inodes on the `mkfs` command line, you can override the more conservative number of inodes that `mkfs` calculates for you otherwise. You can even request 65,535 inodes, creating a superblock large enough to support the maximum number of files that one file system can contain. (The `du` command can be used

to determine how many inodes and blocks are free.) This can be useful if you know your application will require great numbers of small files (so that partition space is not likely to run out before superblock inode storage runs out).

You may need to create A/UX file systems when you normalize a partition or reconfigure your partitions by resizing, adding, or deleting partitions.

4.2 When reconfiguring partitions

The following is a list of overall steps you should complete to reconfigure partitions or to normalize them:

1. Make backups of your existing file systems.
2. If you know you are going to create file systems of the same size when you partition the disk over again, you can use a file system utility such as `dump.bsd(1M)`. (This utility creates a backup that, when it is restored, overwrites the old superblock as well as the old files.) Otherwise, use a file-oriented backup utility such as `tar(1)` or `cpio(1)`. These two utilities are more flexible, since they allow restoration of individual files or all the files onto any valid A/UX file system with enough space.
3. Use either the procedures in "How to Create an A/UX File System of Maximum Size" or the procedures in "How to Create Custom Partitions."
4. Restore the original contents of the file system using the appropriate utility.

4.2.1 When re-initializing an error-prone disk

Another possible context in which you may need to create an A/UX file system is when, after trying all the less drastic efforts to correct disk errors, you resort to HD SC Setup to re-initialize an error-prone disk. When you normalize partitions (recreating them under HD SC Setup 2.0) you can use the Tape Backup 40 SC hardware to make tape backups of A/UX data as well as Macintosh data. The following is a list of overall steps you take to re-initialize an error-prone disk.

1. Use the short procedure described in "Ensuring Apple HD Compatibility with A/UX."

- ☐ If you don't recover normal operations after the first try, then you don't need to repeat it. Otherwise, if you no longer experience errors, this action alone must have been successful in restoring normal disk operation. In that case, do not perform any of the subsequent steps.

2. If you are still experiencing disk troubles, use the file system consistency check, `fsck(1M)`.

Normally, `fsck` can fix most data storage inconsistencies on the disk, but you may have to run it few times before it no longer reports errors.

3. If you are still experiencing disk troubles, you may have no choice but to re-initialize the disk. Use either the procedures in "How to Create an A/UX File System of Maximum Size" or the procedures in "How to Create Custom Partitions." Take care to make file systems large enough to recover your file systems in their entirety (consult your previous hard-copy records for your old file system sizes).
4. Restore the original contents of each of the file systems using the most recent backups available.

5. How to create an A/UX file system of maximum size

The following procedure lists the steps you take to create an A/UX file system of maximum size within an Apple hard disk SC. This procedure erases any data stored on the disk. If you are using a disk that already contains data, refer to "The General Steps in Creating A/UX File Systems" for important instructions. The disk you are preparing should already be installed physically, along with the associated cables and the SCSI cable terminator. The power should be switched on.

1. Start or restart your Macintosh II after inserting your Utilities Disk in the floppy drive.
2. Open the Utilities Disk by double-clicking its icon, or by clicking its icon to highlight it, and then choosing Open from the File menu.

drive does not appear, the drive may be switched off or incorrectly connected. Check your power switch and connections.

5. Click Initialize if the disk has not been initialized before.

A dialog box appears, reminding you that initializing erases all of the information on the hard disk. To continue, click Init.

Messages appear in the HD SC Setup dialog box, explaining the progress of the initialization. Initialization takes several minutes, depending on the size of your hard disk.

In addition to preparing the disk, HD SC Setup's Initialize command automatically tests the disk, installs the driver, and installs one large Macintosh partition.

If you see a message that the hard disk failed to initialize properly, try again. If you are still unable to initialize, there may be a hardware problem. Check that the cable and power cord are properly connected and also that a terminator is properly installed. If the disk still fails to initialize, see your authorized Apple dealer or representative.

6. If necessary, name the disk.

Unless the disk is already named, the program prompts for a name. You can type up to 27 characters, and you can use any keyboard character except the colon (:). Then press RETURN or click OK. After the disk has a name, a message tells you that initialization was successful.

The disk name appears with the hard disk icon as long as you leave a Macintosh partition on the disk. If you place A/UX partitions exclusively on the auxiliary hard disks, no hard disk icons appear on the desktop when you start the system with a system disk (or when you start the system from the A/UX distribution disk, but quit the `sash` application rather than launch A/UX).

7. Click Partition.

The Partition dialog box appears. See Figure 5-8.

- ☐ Click Remove.

An alert box asks you to confirm that you want to erase the information in the partition.

- ☐ Click OK.

When you remove a partition, the space it occupied becomes gray to represent free space.

10. Add an A/UX partition.

- ☐ Drag the pointer to the gray rectangle representing free space and click in that space.

To create one large partition, adjust the brackets so that they engulf the entire length of the gray free space area. If you move the pointer to the left or the right of the rectangle, the brackets disappear.

The size, in kilobytes, is shown on the left. You don't need to be exactly at the maximum size at this point. You still have a chance to change it.

Apple HD SC Setup immediately presents the Partition Type dialog box, as shown previously in Figure 5-11. You use the left side to select a partition type for the custom partition you are creating, and the right side to adjust its size, if necessary.

- ☐ Select Root or Root&Usr as the partition type by clicking on it in the list at the left of the screen. The default slice number will be 0.
- ☐ If the Adjust Size box shows that your partition is not at the maximum size, double-click the Adjust Size box and enter the maximum size, which is displayed underneath this box.
- ☐ Write down the maximum size parameter.

Doubling this number gives you the number of 512-byte disk blocks that this partition consumes, which will be needed in a subsequent step.

- ☐ Click OK.

C

C

C

The `mkfs` utility constructs a file system on a device such as an HD SC. The format of the `mkfs` command you will enter is

```
mkfs /dev/rdisk/cnd0s0 number-of-blocks
```

The variables in this command for which you must supply values are as follows:

n The SCSI ID number of the disk. This number is normally 5 for the first external HD SC, or 6 for the second.

number-of-blocks The number of physical blocks partitioned on the disk. Calculate this number by doubling the number of kilobytes that HD SC Setup reported as the maximum partition size in step 10.

☐ Replacing the correct values for the variables in

```
mkfs /dev/rdisk/cnd0s0 number-of-blocks
```

enter the `mkfs` command. For example, suppose that the SCSI ID number for the disk you are preparing is 6, and you had noted 42099K as the number of physical blocks available for the partition. If so, after doubling the number of blocks you would enter the following command:

```
mkfs /dev/rdisk/c6d0s0 42099
```

The following message appears:

```
Mkfs:/dev/dsk/c6d0s0?
```

(interrupt if wrong)

You have 10 seconds to signal an interrupt to cancel the `mkfs` command. Unless you have made a typing error, don't interrupt the command. The `mkfs` command begins constructing a file system on the partition of your external HD SC. When `mkfs` is finished, it reports the total number of logical blocks and inodes created on the disk and returns your command prompt. This may take a few minutes.

15. Run `fsck` on the new file system partition.

To ensure the consistency of the file system you just made, run the `fsck` utility and enter the following command, but be sure to substitute the hard disk's SCSI ID number for *n*.

```
fsck -y /dev/rdisk/cnd0s0
```

For example, if the SCSI ID number of your new external HD SC is 6, use the following command:

```
fsck -y /dev/rdisk/c6d0s0
```

If there are any inconsistencies, `fsck` automatically repairs them for you. If you prefer to see the inconsistencies and confirm repair, do not use the `-y` flag.

16. Decide on a mount point for the new file system.

You attach an A/UX file system to the rest of the directory hierarchy at a mount point. You can use any directory for a mount point. However, it is usually best to use an empty directory, because the mounted file system overlays any files in the directory, making them inaccessible during that time. The empty directory `/mnt` was shipped on your A/UX distribution disk as a convenient mount point. However, you may want to reserve `/mnt` for temporary file systems; for example, you may wish to use `/mnt` for mounting file systems on removable 3.5-inch disks.

To create the directory intended for use as a mount point, enter:

```
mkdir /mount-point
```

For the example in the next few steps, slice 0 of your external HD SC with SCSI ID number 6 is assumed to be dedicated to hold documentation. In that case, the following command line makes a descriptive mount point:

```
mkdir /pubs
```

This directory `/pubs` is the mount point where you could attach the newly created file system.

17. Mount the new file system.

Replacing the SCSI ID number of the HD SC for *n*, and replacing the partition slice for *y*, enter the following command:

```
mount -v /dev/dsk/cnd0sy /mount-point
```

The command returns a message informing you that your new file system is now mounted at *mount-point*.

For the example cited earlier, you would enter

```
mount -v /dev/dsk/c6d0s0 /pubs
```

18. Create a *lost+found* directory for the new file system.

- ☐ Change the directory to the mount point of the new file system.

```
cd /mount-point
```

For the example cited earlier, enter the command

```
cd /pubs
```

- ☐ Type the following command to create a *lost+found* directory on the new file system:

```
mklost+found
```

This command reports its progress as it creates a *lost+found* directory in the top directory of the new file system, and creates and removes empty files to make slots available for future use.

In the future, when you run *fsck* on this file system after a system crash, and the files have become unconnected in its directory structure, *fsck* will place the files in this *lost+found* directory. Files connected to *lost+found* are named by their inode number instead of their original filenames. You can use the *ls-l* command to determine who the owners are, and then tell the owners

to copy the files back to their directories under their original filenames.

19. Test the file system.

To verify that your new file system is accessible, try writing to it and reading from it with the following commands:

```
cp /etc/passwd /mount-point/mount.test
cat /mount-point/mount.test
```

Your system's password file should scroll across your screen. Enter the command

```
rm /mount-point/mount.test
```

to remove the test file.

If the password file did not appear, reenter the commands in this step and make sure you type them correctly. If the password file still does not appear, start over at step 14.

20. Update the mount entries in `/etc/fstab`.

You'll also want to update the file system table in `/etc/fstab`.

When changing an important system file like this, it is always a good idea to first make a copy of it. That way, if something goes wrong, you can always reinstate the copy to its original name and restore your system to its previous state. After copying `/etc/fstab`, edit the file to include the line

```
/dev/dsk/cnd0s0 mountpoint 5.2 rw 0 2
```

For an explanation of the fields in `/etc/fstab`, see Figure 8-8, "A Description of Sample Entries in `/etc/fstab`." See the section "Multiple File Systems and `fsck`" in Chapter 8 for more information on the relevance of these fields. You can also see `fstab(4)` in *A/UX Programmer's Reference*.

6. About creating custom partitions

HD SC Setup may be used with any Apple SCSI hard disk—such as the Apple HD 20SC, the HD 40SC, or the Apple HD 80SC—that's

connected internally or externally to a Macintosh computer. HD SC Setup provides A/UX users with an easy way to allocate space for custom-size A/UX partitions.

The following procedure leads you through the use of HD SC Setup to create a custom disk partitioning scheme, which is recorded in the disk's partition map (see "Why Disk Subdivisions Are Beneficial" for details).

At step 4 of this procedure, you are instructed to initialize the disk if necessary. This will destroy any existing information on the disk. If the disk contains data, the disk is already initialized, and you need not reinitialize it. Likewise, the partitioning that you do after initialization will render the old data unreadable (unless the new partitioning scheme is exactly the same as the old one and you used the same version of HD SC Setup you used to create the new and original partitions). If you are using a disk that already contains data, refer to "When Reconfiguring a Partition" for important instructions. The only time you should re-initialize is when the information on your disk has become unreadable, and only after you have made repeated attempts to fix it using `fsck(1M)` and HD SC Setup as described in "When Reconfiguring a Partition."

The sequence of steps you take to create a custom partition scheme flows predictably until the final step of "How to Create a Custom Partition." Then the course of action you take depends on the partitioning scheme you are trying to create.

6.1 How to create a custom partition

Follow steps 1-8 in "How to Create an A/UX File System of Maximum Size." Then follow these steps to create custom partitions:

9. Use the subprocedures in "Subprocedures for Creating Custom Partitions" as necessary to create the desired partitions.

At this step, you must decide what subprocedures detailed in "Subprocedures for Creating Custom Partitions" are necessary to arrive at the desired number of partitions and desired sizes. These additional instructions lead you through the use of the Details, Remove, and Group buttons. There are also instructions for the addition of a new partition, which is invoked by clicking in the gray area representing free space.

If there is no free space in which to create new partitions (there is no gray area), you may need to remove a partition, such as the Macintosh partition that is automatically created when you initialize the disk. You can start adding custom-size partitions in this newly gained space.

To resize an existing partition, you must first remove it, then recreate it at the desired size (see "To Remove a Partition" and "To Add a Partition").

10. When you have created the desired partitioning scheme, turn to "Quitting HD SC Setup."

7. Subprocedures for creating custom partitions

From "How to Create Custom Partitions," you were directed to proceed to the subprocedures that follow. At this point, you can proceed in various ways, depending on the number and sizes of the partitions you wish to create. The subprocedures that you may need are those for removing, adding, grouping, moving, and viewing partitions. Each of these subprocedures is described in a separate section.

After you have created the custom partitions you desire, turn to "Quitting HD SC Setup."

7.1 To remove a partition

You remove existing partitions to make space for new custom partitions. Working in the Custom Partition dialog box, you can remove any partition from your hard disk.

The following steps lead you through the removal of a partition. Before you can use this subprocedure, you must have on the screen the Custom Partition dialog box shown previously in Figure 5-9.

Important: Don't remove the driver partition unless you have a special reason. Without the driver partition, you won't be able to use your disk after restarting your Macintosh.

1. Select the partition you wish to remove by clicking anywhere in its rectangle.

The name of that partition is highlighted to show that it has been selected.

2. Click Remove.

An alert box asks you to confirm that you want to erase the information in the partition.

3. Click OK.

Click Cancel if you decide not to remove the partition.

When you remove a partition, the space it occupied becomes gray to represent free space. If another area of free space is adjacent, the two rectangles are combined.

7.2 To add a partition

Before you can add a new partition, there must be a section of free space large enough to hold your new partition. If there is insufficient free space, remove one or more partitions. (For more information, see the preceding section, "To Remove a Partition.")

If the free space is divided into sections by existing partitions, with no single section large enough to hold your new partition, you need to remove a partition or combine the sections of free space by *grouping* the partitions. (For more information, see the next section, "To Group Partitions.")

The following steps lead you through the addition of a new partition. Before using this subprocedure, you must be at the Custom Partition dialog box shown in Figure 5-9.

1. Drag the pointer to a gray rectangle representing free space, and click in that space.

As shown previously in Figure 5-10, HD SC Setup draws two brackets representing the new partition. If you place the pointer in the upper half of the free space rectangle, the brackets start at the top of the free space; if you place the pointer in the lower half of the free space rectangle, the brackets start at the bottom of the free space.

2. Drag the pointer up or down to adjust the size of the new partition.

If you move the pointer to the left or the right of the rectangle, the brackets disappear.

The size, in kilobytes, is shown on the left.

3. Release the mouse button when you are satisfied with the size of the new partition.

You don't need to be exact. You still have a chance to change it.

Apple HD SC Setup immediately presents the Partition Type dialog box, as shown previously in Figure 5-11. You use the left side to select a partition type for the custom partition you are creating. You use the right side to adjust its size.

4. Select the partition type by clicking in the list on the left.

You have several choices for A/UX partitions, some of which are automatically assigned a slice number for use under A/UX, as Table 5-1 shows.

Table 5-1. A/UX partition types available

A/UX Partition Type	A/UX Slice Number
Eschatology, Eschatology2	(none)
Misc A/UX	(none)
Root, Root&User	0
Swap	1
Usr	2

Note that three of the A/UX partitions are not automatically mapped into a slice number. They are Eschatology, Eschatology2, and Misc A/UX. To associate these partitions with slice numbers, you must use the A/UX pname(1M) utility (see "Using dp and pname").

5. If you wish to change the size of the partition, double-click the Adjust Size box and enter the correct size in kilobytes.

You can enter the partition size to a precision of a half-kilobyte (0.5K). HD SC Setup will not allow other fractions.

You can change the size by typing a new number for the size of the partition.

The maximum possible size is shown below the Adjust Size box; if you've selected a partition type, the minimum possible size is also shown.

6. Click OK.

The HD SC Setup program creates the new partition and again presents the Custom Partition dialog box, where the new partition is shown.

You can create another custom partition as long as you have sufficient free space, but you can select only a listed partition type.

Click Cancel in the Partition Type dialog box if you wish to return to the Custom Partition dialog box without creating a new partition.

7. Click Done in the Custom Partition dialog box to return to the main HD SC Setup dialog box.

7.3 To group partitions

If you have created two or more partitions on your disk, and free space separates them, you may eventually end up with multiple free space areas that could be consolidated into one large free space area.

Grouping partitions combines the free space on your disk.

Sometimes this function is not available, for instance, when there are no free space areas remaining. At such times the Group button is dimmed.

Before you can use this subprocedure, the Custom Partition dialog box, shown in Figure 5-9, must be on the screen.

1. Click Group.

HD SC Setup presents an alert box, warning that moving information from one portion of your disk to another will take time. Because grouping usually means that a large amount of information is being moved, HD SC Setup also warns that some information might be lost.

2. Click OK.

All partitions are grouped together on the disk, and they are shown together at the top of the Custom Partition display.

Click Cancel if you decide not to group the partitions.

7.4 To move a partition

You can also use the mouse to move a partition into adjacent free space or into any free space larger than the partition.

Before you can use this subprocedure, the Custom Partition dialog box, shown in Figure 5-9, must be on the screen.

1. Click the partition to be moved.
2. Drag the partition to its new position.

HD SC Setup won't let you move a partition "just a little bit" into an adjacent free space. You must drag the partition more than half way.

When you release the mouse button, HD SC Setup presents an alert box, warning that moving information from one portion of your disk to another will take time. HD SC Setup also warns that some information might be lost in the process.

3. Click OK to confirm.

Click Cancel if you decide not to move the partition.

7.5 To view your partitions

Before you can use this subprocedure, the Custom Partition dialog box, shown in Figure 5-9, must be on the screen.

Follow these steps to see the size of your partitions:

1. Click the Details button.

Select Restart from the Special menu. Do not reinsert the Utilities disk. Rather, let the system boot from your installed version of A/UX. The `sash` startup application should boot A/UX to single-user mode.

2. Use `mkfs` to make a file system inside an existing partition.

The `mkfs` utility constructs a file system on a block device such as an HD SC. The format of the `mkfs` command you will enter is

```
mkfs /dev/dsk/cnd0sy number-of-blocks
```

The variables for which you must supply values in this command are:

- n* The SCSI ID number of the disk. This number is normally 5 for the first external HD SC, or 6 for the second.
- y* The A/UX disk slice number, which identifies a particular partition.

number-of-blocks The number of physical blocks that `mkfs` should reserve for the new file system. You can arrive at this number by doubling the number of kilobytes that were allocated for the corresponding partition within HD SC Setup, or by noting the number of physical blocks reported for each partition name after entering

```
echo P | dp -q /dev/dsk/cnd0s31 | egrep "Name|Phy"
```

The resulting report shows the name and size of each partition, for example,

2. Click OK to close the Details window and return to the Custom Partition dialog box.

8. Quitting HD SC Setup

1. Click Done in the Custom Partition dialog box to return to the main HD SC Setup dialog box.

When you are satisfied with the partitioning scheme you have obtained using the subprocedures, click Done.

2. Click Quit to return to the desktop.

No icon will show on the desktop for the new disk unless a Macintosh partition was created somewhere on the disk .

3. Proceed to the section, "Making an A/UX File System Partition."

9. Making an A/UX file system partition

You probably want to store files and programs in the partitions created by HD SC Setup. To do so, you first need to create the A/UX file system structures that support them. Another possible use for partition space is as an A/UX swap area, as described in "Adding Swap Space." Before proceeding with an A/UX file system partition, you should know how A/UX references partitions.

9.1 Making and mounting an A/UX file system

After partitioning your disk with HD SC Setup, you can make A/UX file systems for those partitions in which you intend to store files and programs. For each file system, perform the following steps.

Note: Some partition configurations cannot take advantage of the automatic slice number mapping. Table 5-1 shows how A/UX attempts to map the various partition types to unique slice numbers. Until you associate these partitions with slice numbers manually, you cannot use them in the following procedure. Refer to the next section, "Using dp and pname," as necessary before continuing.

1. Boot A/UX to single-user mode.

that the SCSI ID number for the disk you are preparing is 6, you created a root partition at slice 0, and the `dp` utility reported that your disk has 156353 physical blocks available for the partition. If so, you would enter the following command:

```
mkfs /dev/dsk/c6d0s0 156353
```

The following message appears:

```
Mkfs: /dev/dsk/c6d0s0?
```

(interrupt if wrong)

- ☐ You have 10 seconds in which to signal an interrupt to cancel the `mkfs` command. Unless you have made a typing error, don't interrupt the command. The `mkfs` command begins constructing a file system on the partition of your external HD SC. When `mkfs` is finished, it reports the total number of logical blocks and inodes created on the disk and returns your command prompt.

3. Run `fsck` on the new file system partition.

To ensure the consistency of the file system you just made, run the `fsck` utility by entering the following command, but be sure to substitute the SCSI ID number of the HD SC for *n* and to substitute the slice number of the partition for *y*.

```
fsck -y /dev/dsk/cnd0sy
```

For example, if your new external HD SC has SCSI ID number 6 and the slice number of the partition is 0, enter the command

```
fsck -y /dev/dsk/c6d0s0
```

Name: "Swap", Type: "Apple_UNIX_SVR2"

Physical: 19606 @ 108532, Logical: 19606 @ 0

You can associate these physical sizes with the correct slice number through the partition names. Thus, you can associate the swap partition reported previously with the line identified as the swap partition by entering

`pname -p`

Each line reported by `pname` includes the partition name and type, followed by three numbers separated by colons. The last of these numbers is the slice number. (These three numbers correspond to the variables in the pathname `/dev/dsk/cndmsy`.)

`A/UX Root:Apple_UNIX_SVR2:0:0:0`

`Swap:Apple_UNIX_SVR2:0:0:1`

For the sample output shown in the preceding examples, you can see that a partition named `Swap` is slice 1 of disk 0, and that its size is 19606 physical disk blocks.

Note: If the report created with `pname -p` does not include a partition that you made with HD SC Setup, you can still associate it with an A/UX slice number in one of two ways: (1) If the partition type is `Root` (or `Root&Usr`), `Swap`, or `Usr`, use slice numbers 0, 1, or 2, respectively. (2) Otherwise, use `dp` and `pname` to force A/UX to recognize the partition and provide it with a slice number, as described in the next section, "Using `dp` and `pname`." The latter method is also necessary when you have more than one partition of the same type, such as two swap partitions. It is also necessary when you have more than one partition of either `Root` or `Root&Usr` type.

- ☐ Replacing the correct values for the variables in

`mkfs /dev/dsk/cnd0sy number-of-blocks`

enter the `mkfs` command. For example, suppose

If any inconsistencies exist, `fsck` automatically repairs them for you.

4. Make a mount point for the new file system.

You attach an A/UX file system to the rest of the directory hierarchy at a mount point. You can use any directory for a mount point, but any files in the directory will be obscured for the duration of the mount. For example, the empty directory `/mnt` was shipped on your A/UX distribution disk as a convenient mount point. However, you may want to reserve `/mnt` for temporary file systems; for example, you may wish to use `/mnt` for mounting file systems on removable 3.5-inch disks.

To create the directory intended for use as a mount point, enter

```
mkdir mount-point
```

For the example in the next few steps, slice 0 of your external HD SC with SCSI ID number 6 is assumed to be dedicated to hold documentation. In that case, the following command line makes a descriptive mount point:

```
mkdir /pubs
```

This directory `/pubs` is the mount point where you could attach the newly created file system.

5. Mount the new file system.

Replacing the SCSI ID number of the HD SC for *n*, and replacing the partition slice for *y*, enter the following command:

```
mount -v /dev/dsk/cnd0sy mount-point
```

The command returns a message informing you that your new file system is now mounted at *mount-point*.

For the example cited earlier, you would enter

```
mount -v /dev/dsk/c6d0s0 /pubs
```

6. Create a `lost+found` directory for the new file system.

- ☐ Change the directory to the mount point of the new file system.

```
cd mount-point
```

For the example cited earlier, enter the command

```
cd /pubs
```

- ☐ Enter the following command to create a `lost+found` directory on the new file system:

```
mklost+found
```

This command reports its progress as it creates a `lost+found` directory in the top directory of the new file system, and creates and removes empty files to make slots available for future use.

In the future, when you run `fsck` on this file system after a system crash, and files have become unconnected in its directory structure, `fsck` will place the files in this `lost+found` directory. Files connected to `lost+found` are named by their inode number instead of their original filenames. You can use the `ls -l` command to determine who the owners are, and then tell the owners to copy the files back to their directories under their original filenames.

7. Test the file system.

To verify that your new file system is accessible, try writing to it and reading from it by giving the following commands:

```
cp /etc/passwd mount-point/mount.test
cat mount-point/mount.test
```

Your system's password file should scroll across your screen. Enter the command

```
rm /mount-point/mount.test
```

to remove the test file.

If the password file did not appear, reenter the commands in this step and make sure you type them correctly. If the password file still does not appear, start over at step 1 of this section and repeat all of the steps.

8. Update the mount entries in `/etc/fstab`.

You'll also want to update the file system table in `/etc/fstab`.

When you change an important system file like this, it is always a good idea to make a copy of it first. That way, if something goes wrong, you can always reinstate the copy to its original name and restore your system to its previous state. After copying `/etc/fstab`, edit the file to include the line:

```
/dev/dsk/cnd0s0 mountpoint 5.2 rw 0 2
```

For an explanation of the fields in `/etc/fstab`, see Figure 8-8, "A Description of Sample Entries in `/etc/fstab`." See the section "Multiple File Systems and `fsck`" in Chapter 8 for more information on the relevance of these fields. You can also see `fstab(4)` in *A/UX Programmer's Reference*.

9.2 Using `dp` and `pname`

Perform the following procedure if you have made a partition configuration that:

- includes any partitions of type Eschatology, Eschatology2, or Misc A/UX

- includes two or more partitions that map to the same slice number (For example, Root and Root&Usr both map to slice 0.)

These kinds of partitioning configurations cannot take advantage of the automatic slice number mapping. (Table 5-1 shows how A/UX attempts to map the various partition types to unique slice numbers.)

Until you associate these partitions with slice numbers, you cannot use them for A/UX purposes, such as for creating, mounting, and checking file systems.

The following procedure leads you through the necessary steps to locate partitions with non-unique slice numbers or missing slice numbers, and to make any necessary changes. First, you should have already created and mounted file systems with automatically mapped slice numbers.

1. Boot A/UX to single-user mode, if you have not done so already.

Select Restart from the Special menu. Do not reinsert the Utilities disk. Rather, let the system boot from your installed version of A/UX. The startup application should boot A/UX to single-user mode.

2. Use the `dp` utility to obtain the index numbers for any partitions with identical names, and if present, change the duplicates to unique names.

- ☐ Replacing the drive number *n* with the SCSI ID number of the desired disk, enter the following command:

```
echo P | dp -q /dev/rdisk/cnd0s31 | egrep "Index/Name"
```

The `fstab` file includes all the old mount points. The `dp` utility responds with a report showing the index number and name of each partition allocated on that disk using the Apple HD SC Setup program. Note these values. You will use them in a subsequent step.

The names shown by `dp` are usually very close to the partition types you saw in HD SC Setup. One exception is that the Misc A/UX partition type is named Random A/UX fs under `dp`.

If duplicate names exist, you must assign unique names by using the editing capabilities of `dp`. If no duplicates exist, proceed to step 4.

- ☐ Replacing the drive number *n* with the SCSI ID number of the new HD SC, enter the following command:

```
dp /dev/rdisk/cnd0s31
```

The `dp` utility then prompts you for a command. For example, if you are partitioning a disk with SCSI ID number 6, the following message appears:

```
"/dev/dsk/c6d0s31" n partitions, m allocated [unknown sizes]
Command?
```

Note: If you get a different response, press the lowercase `q` to quit `dp`. Reenter the `dp` command just given, making sure that you specify the correct SCSI ID and that you type the command correctly.

- ☐ Repeat the following series of substeps as many times as necessary to eliminate duplicate partition names.

- Replacing the value *x* with the index number of one of the identically named partitions, enter

```
cx
```

You are prompted to identify the attribute field that you wish to change.

```
DPME Field?
```

- Enter

n

to begin changing the partition's *name* field.
Now you're prompted for a name for the partition.

Name [*Old-name*]:

- Give the partition a name that is unique and that does not contain any embedded spaces. If A/UX_Partition has not already been used, then you can enter

A/UX_Partition

Again you're prompted for the attribute field that you wish to change.

DPME Field?

- Enter

q

to return to the dp utility's first menu level. You will see the Command? prompt.

Command?

- ☐ If you have no more non-unique partition names, you should save all the accumulative changes and quit dp by entering:

w

q

The root command prompt should reappear on your screen.

Otherwise, if you still have any remaining partitions without unique names, return to the substep in which you enter the index number (cx).

3. If you have partitions named other than Swap, Root or Root&User, or Usr, then use `pname` to associate slice numbers with these unmapped partitions.

Using the `-a` flag causes `pname` to write the selected slice to the file `/etc/ptab`. Once this is done, the appropriate association will be made each time you boot A/UX.

- ☐ Replacing the drive number *n* with the SCSI device number of the disk, and *Name* with the name of a remaining unmapped partition, enter

```
pname -a -cn "Name"
```

at the root command prompt. The `pname` utility responds with the A/UX pathname by which that partition can now be referenced for general use. Note that `pname` has assigned a slice number for the partition, and so the partition is no longer unmapped.

- ☐ If you have no remaining unmapped partitions, proceed to the next step; otherwise, repeat the `pname` command for every unmapped partition.
4. If you have both a Root and a Root&User partition, their automatically mapped slice numbers won't be unique. Use `pname` to give one a new slice number.

- ☐ To discover which of the two conflicting partitions is currently recognized, enter

```
pname -p
```

at the root command prompt.

- ☐ Next, let `pname` assign a new slice number to the currently unrecognized partition. Enter the name of the partition not yet recognized.

```
pname -a -cn "Name"
```

The `pname` utility responds with the A/UX pathname by which that partition can now be referenced for general use.

5. If you want to run file system consistency checks against the new partitions each time you reboot, then the startup files need to be altered. See "Multiple File Systems and `fsck`" in Chapter 8.

10. Adding swap space

You can use the space allocated for a partition not only to create file systems but also to increase the A/UX swap area. The partition that you wish to use for swap space must already be associated with a unique slice number. Refer to the previous section, "Using `dp` and `pname`," to determine whether you need to perform any additional steps and what those steps are. These A/UX requirements for disk access are explained in detail in "The Three Steps of a File Access."

Two steps are involved in using a partition as additional swap space. The first is to obtain the information you will need to specify in the `/etc/swap` command line. Once you know these details, the next step is to enter the appropriate command request containing those details.

1. To start, obtain the starting disk block where the partition begins and the length of the partition in disk blocks. Replacing the SCSI device number *n* with the correct value, enter

```
echo P | dp -q /dev/dsk/cnd0s31 | egrep "Name/Phys"
```

Note the statistics provided for the partition you wish to be used as swap space. For example, if the partition name is `swap`, the block offset to it is 108532, and its size is 19606, the following two lines of information will appear somewhere in the output of the preceding command:

```
Name: "Swap", Type: "Apple_UNIX_SVR2"  
Physical: 19606 @ 108532, Logical: 19606 @ 0
```



```
pname -cn "Name"
```

If you run the preceding `pname` command, a pathname is returned. It shows what slice number is associated with the partition in terms of the standard pathname format `/dev/dsk/cnd0sy` (where *y* is the slice number).

Otherwise, if you did not have to map in a slice number yourself, the slice number is reported as the last of the three digits at the end of each line of output from the original `pname` command. The first of the three digits is the SCSI device number, which can help you identify the correct partition when you have two partitions with identical names on different disks. For example, assume the desired swap partition is on SCSI device 5. Then the following sample output shows that the correct slice number is 4:

```
Swap:Apple_UNIX_SVR2:0:0:1
```

```
Swap:Apple_UNIX_SVR2:5:0:4
```

4. Now you have all the important information you need for actually increasing your swap space. Replacing the SCSI device number for *n*, slice number for *y*, the offset for *offset* (108532 for the example cited earlier), and the physical size for *length*, enter

```
/etc/swap -a /dev/dsk/cnd0sy offset length
```

To confirm that you have done what you set out to do, obtain a report of the swap spaces currently in use by entering

```
/etc/swap -l
```

One-line descriptions of the swap areas are reported for each swap area currently recognized.

PLACE TAB
HERE

MANAGING DISKS

BACK OF TAB

BLANK PAGE

Chapter 6

Managing Disks

Contents

1. Introduction	1
2. About autorecovery	1
2.1 Overview	2
2.2 Using autorecovery	2
2.3 How autorecovery works	3
2.4 autorecovery administration	4
2.4.1 The <code>eu</code> utility	4
2.4.2 The <code>escher</code> utility	5
2.4.3 The <code>eupdate</code> utility	5
2.4.4 Administration guidelines	6
2.4.5 Troubleshooting	6
3. Reclaiming disk space	12
3.1 Trimming files that grow	12
3.2 Serving read-only files via NFS	14
3.3 Compressing infrequently used files	15
3.4 Usage notes	16
3.4.1 Extensions to names of compressed files	16
3.4.2 Compressing an archive of files	16
3.5 Automating system administration with <code>cron</code>	17
4. AppleCD-ROM and A/UX	18
4.1 Mounting a CD-ROM as an A/UX file system	19
4.2 Mounting remotely	20

Chapter 6

Managing Disks

1. Introduction

Many sizes and makes of hard disks are available for use with A/UX. Regardless of make or size, any disk can become full, requiring you to provide more disk space. Also, any disk may fail, requiring you to recover data rather than lose it. This chapter gives suggestions on various means to free up space on disks to make room for user files. This chapter also describes **autorecovery**, a feature unique to A/UX. If you engage in periodic maintenance, you can use **autorecovery** to rebuild your system after a system crash. This chapter details the steps you take to use **autorecovery** successfully.

You can install an Apple compact disc read-only memory (CD-ROM) under A/UX, thereby increasing storage capacity greatly. This chapter tells you how to install Apple's CD-ROM both on a single system and over a network.

2. About autorecovery

The **autorecovery** program is designed to ensure that an A/UX system can be brought to multi-user mode, capable of operating as part of a network. Before the system is booted, **autorecovery** identifies and compensates for bad disk blocks, file-system inconsistencies, and missing or damaged files. The **autorecovery** program is designed to protect you from sudden, catastrophic loss of data and to minimize the need for a technical expert to diagnose and repair system problems.

The **autorecovery** program does have limitations: It is concerned only with critical system files, and it does not restore damaged or missing user files. You must keep backup copies of your own files in case you need to restore them.

Although most of **autorecovery**'s operation is automatic, you must perform some administration tasks to keep **autorecovery** running smoothly. This section describes the administration tasks.

2.1 Overview

The terms *autorecovery* and *eschatology* refer to the same A/UX feature. The term *autorecovery* refers to both the procedure that checks and repairs the A/UX file systems and to the special disk file systems used by this procedure. Eschatology is used in filenames and command names and in the arguments to commands.

Two areas on the A/UX disk are reserved for *autorecovery*. Each area is a distinct A/UX file system that contains copies of key system files and other information about A/UX. If a key system file is damaged or destroyed, *autorecovery* copies the file from one of these file systems to the A/UX root file system.

The *autorecovery* program uses a list of key system files known as the configuration master list, or CML. The CML appears in the A/UX root file system in the file `/etc/eschatology/init2files`. A copy of this file appears in each *autorecovery* file system.

At boot time, *autorecovery* verifies the physical condition of the disk and then checks each file in the CML. The *autorecovery* program uses rules stored in the CML to check file attributes, such as size, ownership, permissions, type, modification time, version, and checksum. If any attributes do not match, *autorecovery* corrects the file attributes, if possible. If *autorecovery* cannot make these corrections, it replaces the file.

Because *autorecovery* depends on the CML, you must keep the CML up to date. When you add or change key system files, you must add new entries to the CML and update the files in the *autorecovery* file systems, using the *autorecovery* utilities `eu` and `escher`. The *autorecovery* program cannot function well unless a conscientious system administrator keeps the CML and *autorecovery* file systems updated.

2.2 Using *autorecovery*

The *autorecovery* program is run from the stand-alone shell (`sash`) when the system is booted. (See Chapter 2 of this book, "System Startup and Shutdown," and `sash(8)` in *A/UX System Administrator's Reference* for information on `sash`.) You can either run *autorecovery* manually each time you boot or set it up to run

automatically. To run autorecovery manually from `sash`, enter the command

```
esch -v -b
```

For a detailed explanation of the `esch` command and its options, see `esch (8)` in *A/UX System Administrator's Reference*. To set up the system to run autorecovery each time you boot, follow these steps:

1. Select the Cancel option from the `sash` window.
2. Select the Booting option from the Preferences menu.
3. Enter this command in the Autorecovery box:

```
esch -v -b
```

4. Click OK.

To suppress the automatic running of autorecovery, follow the same steps, but instead of entering the `esch` command in the Autorecovery box, enter `echo no autorecovery`.

2.3 How autorecovery works

The autorecovery program proceeds through several phases. First, it examines the system information in its own file systems to verify that a system failure did not interrupt the previous invocation of autorecovery. If the autorecovery file systems are suspect, autorecovery cannot use them to restore damaged or missing files in the A/UX root file system. If autorecovery detects that an autorecovery file system was being updated when the system failed, it does not use that file system. If both autorecovery file systems are suspect, autorecovery does not continue.

After checking its own file systems, autorecovery checks each A/UX file system individually, verifying that all blocks are readable. It marks bad blocks so that they will not be used.

The autorecovery program then uses a version of `fsck` to check the A/UX root file system and each autorecovery file system for consistency. It attempts to correct any errors it finds. If a file system is not repairable, autorecovery remakes it as a last resort. When

autorecovery remakes a file system, all data in the file system is lost and must be restored from backups.

After all file systems have been checked, autorecovery verifies the A/UX key system files listed in the CML, checking the attributes of each file against the rules specified in the CML. If autorecovery finds inconsistencies, it corrects the file, if possible, or replaces it with a copy from the autorecovery file systems.

The entire autorecovery process takes from 45 minutes to an hour. Most of this time is spent verifying each disk block. You can skip this phase by using the `-b` option of the `esch` command.

Without this phase, autorecovery typically takes about 5 minutes, unless major system damage has occurred. See Chapter 8, "Checking the A/UX File System: `fsck`" for a description of the file system checking routine (`fsck`).

2.4 autorecovery administration

You must perform two key autorecovery administration tasks:

- Update the CML when key system files change.
- Update the autorecovery copies of key system files when they change.

You perform these tasks with the `escher`, `eu`, and `eupdate` utilities, described in this section.

The utilities `escher` and `eu` perform similar functions: both update the CML and the autorecovery file systems, but they use slightly different procedures. The `eupdate` utility copies the system files typically updated by autoconfiguration to the autorecovery file systems and updates the CML entries for these files. Each utility is described next and in *A/UX System Administrator's Reference*.

2.4.1 The `eu` utility

The `eu` utility updates the CML and copies a specified file to the autorecovery file systems. The `eu` utility operates on only one file at a time. To run `eu`, enter the command

`eu pathname`

where *pathname* is the complete pathname of the file to be copied. A

complete pathname always begins with a slash (/).

The `eu` utility has no interactive mode. As a rule, you should use `eu` to update the CML, because it creates entries with a checksum rule. The `eu` utility updates the CML even if the CML already contains an entry for the specified file.

You must be the root user to run `eu`.

2.4.2 The `escher` utility

The `escher` utility also adds new entries to the CML and copies the file to the autorecovery file systems. To run `escher`, enter the command

```
escher pathname
```

where *pathname* is the complete pathname of the file to be added. A complete pathname always begins with a slash (/).

The `escher` utility examines the specified file and adds information about the file to the CML. As discussed in `escher(1M)`'s man page, `escher` does not use all possible CML rules when creating an entry; in particular, entries created with `escher` do not have a checksum rule. If you want a checksum rule, use `eu`. When the CML entry has been created, `escher` copies the file to both autorecovery file systems.

You can also run `escher` interactively. In this mode, `escher` reads the CML and determines which files have been modified more recently than the copies in the autorecovery file systems. For each file that `escher` finds, it prompts you to decide whether the file should be copied to each of the two autorecovery file systems. When the `escher` run is complete, the autorecovery file systems are current with respect to the given files.

You must be the root user to run `escher`.

2.4.3 The `eupdate` utility

The `eupdate` utility updates the CML entries for the files typically modified when you set your system up for networking. To run `eupdate`, enter the command

```
eupdate
```

The A/UX kernel (/unix) and other files necessary for multi-user

network operation are copied to the autorecovery file systems. The CML entries for these files are also updated. You must be the root user to run `eupdate`.

2.4.4 Administration guidelines

To keep your autorecovery file systems current, follow these guidelines:

- Run `escher -m` often, to obtain a list of files that may need to be updated in the autorecovery file systems. You can schedule regular executions of `escher -m` with a `crontab(1)` entry.
- Whenever a key file is added to the system, use the `eu` or `escher` utility to update the CML. The autorecovery program cannot recover files that do not appear in the CML. The file system used for autorecovery has severely limited disk space so be sure files you add are in fact vital to autorecovery's performance.
- Run `eu` or `eupdate` as soon as key system files change. For example, if you run the `autoconfig` command to build a new kernel, you should run `eupdate` as soon as you verify the operation of the new kernel.

You can use the `escher` and `eu` utilities to add entries to the CML. Currently, no utility is available to delete CML entries. Use a text editor for this task. Consult the manual page `cm1(4)` before attempting to update the CML manually.

2.4.5 Troubleshooting

This section discusses problems that occasionally occur when you run autorecovery. Although autorecovery is fairly robust, it errs on the side of caution: autorecovery will not replace damaged or missing files if the copies on the autorecovery file systems or the autorecovery file systems themselves are suspect. This section presents the procedures to use when manual intervention is required to correct the operation of autorecovery.

Note: The procedures outlined here are for emergency use only. Normally, only autorecovery has access to the autorecovery file systems. Manual intervention should be

kept to a minimum. The procedures recommended here use the `pname` utility. Users unfamiliar with this utility should consult the manual page for `pname(1M)` in *A/UX System Administrator's Reference* before continuing.

This section is organized by symptom. Error messages or warning messages may appear as part of the problem description. When appropriate, they are included here.

The following messages may appear when `autorecovery` is run from the `sash` partition.

Symptom	Message
One or both autorecovery file systems have inconsistent mount times	Warning. Inconsistent mount times in bzb.
Autorecovery fails during boot	esch: no consistent type FSTEFBSBZB (ES_BZBS_FSTEFBS) [error occurred in Block Zero Block module]

Occasionally, `autorecovery` may be unable to restore damaged or missing files, usually because autorecovery file systems were left mounted when the system was halted. After detecting this condition, `autorecovery` does not restore files from any autorecovery file systems that were left mounted when the system was interrupted. The `autorecovery` program does not use these file systems because the integrity of the data is not guaranteed. If both autorecovery file systems were mounted when the system was halted, `autorecovery` cannot proceed.

The most obvious sign that `autorecovery` has been interrupted is the message just listed. The `autorecovery` program verifies that

the unmount time is later than the mount time for each autorecovery file system. If an autorecovery file system has been left mounted, this test will fail, resulting in the warning message. The autorecovery file systems that were left mounted when the system was halted must be checked for integrity. Use the following procedure to check the offending file system or systems.

1. Launch `/unix` from `sash` by entering
`launch /unix`
2. When the system prompts you to check the root file system, enter `y`.
3. Bring the system to single-user mode.
4. At the root prompt, enter the command (assuming your root file system has SCSI ID 0)
`pname -s3 "Eschatology 1"`
5. At the root prompt, enter the command
`fsck /dev/dsk/c0d0s3`
The system will check one autorecovery file system for consistency.
6. At the root prompt, enter the command
`mount /dev/dsk/c0d0s3 /mnt`
7. Enter the command
`umount /mnt`
8. Enter the command
`pname -u /dev/dsk/c0d0s3`
9. Enter the command
`pname -s4 "Eschatology 2"`

10. Enter the command

```
fsck /dev/dsk/c0d0s4
```

The system will check the second autorecovery file system for consistency.

11. Enter the command

```
mount /dev/dsk/c0d0s4 /mnt
```

12. Enter the command

```
umount /mnt
```

13. Enter the command

```
pname -u /dev/dsk/c0d0s4
```

14. Enter the command

```
sync;sync;reboot
```

The system will restart. The autorecovery program should now be operational.

The following messages are produced when both `escher` and `eu` are preparing to update the CML at the same time.

Symptom	Message
<code>escher</code> or <code>eu</code> will not run.	Can't lock cml file. <code>eu</code> : Can't lock the fcml. Try again later.

The two utilities cannot be run at the same time, because each must have exclusive access to the CML. Users who receive this message should verify that either `escher` or `eu` is running, but not both. If neither is running, the file `/etc/eschatology/FCML.lock` may be present. After verifying that no processes are attempting to update the CML, you may remove this file.

The following messages are produced when at least one device is still associated with an autorecovery file system.

Symptom	Message
escher will not run.	/dev/dsk/cXXdYYsZZ previously pnamed
	/dev/rdisk/cXXdYYsZZ previously pnamed

To remove the association, use the pname command by following these steps:

1. Enter the command

`pname`

The system will produce a display something like this:

```
/dev/dsk/c0d0s0: "A/UX Root" "Apple_UNIX_SVR2"  
[not in ptab file]
```

```
/dev/dsk/c0d0s3: "Eschatology 1" "Apple_UNIX_SVR2"  
[not in ptab file]
```

```
/dev/dsk/c0d0s4: "Eschatology 2" "Apple_UNIX_SVR2"  
[not in ptab file]
```

```
/dev/dsk/c0d0s31: "Entire Disk" "Apple_UNIX_SVR2"  
[not in ptab file]
```

2. For each device associated with an autorecovery file system, issue the command

`pname -u device-name`

For example, with the example just given, you enter the commands

```
pname -u /dev/dsk/c0d0s3
pname -u /dev/dsk/c0d0s4
```

3. Verify that no devices are currently associated with autorecovery file systems by entering

```
pname
```

again. The system produces a display something like this:

```
/dev/dsk/c0d0s0: "A/UX Root" "Apple_UNIX_SVR2"
                  [not in ptab file]
/dev/dsk/c0d0s31: "Entire Disk" "Apple_UNIX_SVR2"
                  [not in ptab file]
```

The `escher` utility should now run normally.

The following messages are produced when `autorecovery` is run at boot time.

Symptom	Message
"Replaced" files missing after autorecovery runs.	filename was not replaceable

When `autorecovery` determines that a file is invalid, it attempts to replace it with a valid copy from the autorecovery file systems. If the autorecovery file systems do not contain a valid copy of the file, it will not be replaced. Instead, `autorecovery` will remove it, because it has been identified as invalid. This situation is very unlikely, however, since it means that the CML contains an entry for the file, but the file has never been copied to either of the autorecovery file systems. Alternatively, the file may exist on the autorecovery file systems but fail the rules specified in the CML. Consistent use of the `eupdate`, `eu`, and `escher` utilities should prevent this problem from occurring.

The `autorecovery` program removes invalid files before attempting to replace them by design. Since `autorecovery` is concerned only with key system files, damage to these files could pose a security risk, especially in a network environment. For this reason,

autorecovery removes files identified as invalid, whether or not a suitable replacement is available.

When run interactively, `escher` produces the following message.

Symptom	Message
Some files cannot be copied to autorecovery file systems.	Ignoring filename - cmlfile entry incorrect

The `escher` utility complains when the CML entry for the file to be copied contains an invalid rule specification, or when the CML fields are not separated by the correct number of tab characters. Manual editing of the CML is the most frequent cause of this condition.

To recover, first make a backup copy of the CML. Then delete the offending entry from the CML with an editor. Use `escher` or `eu` to restore the entry to the CML and copy the file to the autorecovery file systems.

3. Reclaiming disk space

A/UX includes all the files needed to run A/UX locally. It also includes many added features, plus the source files for an array of public domain software. However, single-disk systems do not have much room available for user data. You can make more room on your disk in several ways. You can remove software that you do not use, for example, the public domain software and the games. If you're on a network, you can place read-only files, such as the on-line manual pages, on a server and remove them from the client machines. You can also condense infrequently used files to reduce the space required for storing them on the disk, and expand them again on demand. It is also prudent for system administrators to trim files (for example, log files) that tend to grow over time.

3.1 Trimming files that grow

As A/UX runs, it creates and adds to assorted log and data files. If no corrective action is taken, these files can easily grow to substantial size. Fortunately, it is easy to monitor the growth of these files, and you can use `cron` to automate much of the task of reducing their size.

Different techniques are needed for different files, however. Some files, such as `core` files, are produced by isolated system events. Once any needed information has been gained from such files, they should be removed. A one-week "cooling-off" period is usually sufficient to ensure that such a file is no longer of interest.

Other files are produced by the system to log events of administrative interest. These files should normally be trimmed of their earlier entries. The `tail` utility is particularly handy for this task.

Finally, some files must be present, but can be truncated to zero length periodically. For instance, the command `cp /dev/null foo` truncates file `foo`. Note that some log files are written into only if they exist. Removing such files causes no error condition but disables the logging activity.

Here is a short list of system files that may need occasional action:

- `/core` — memory images from aborted programs (remove)
- `/etc/wtmp` — record of spawned shells (trim)
- `/dev/*` — accidentally written data files (remove)
- `/lost+found/*` — unlinked files, salvaged by `fsck` (remove)
- `/mnt/*` — misdirected files because of an unmounted file system (remove)
- `/usr/adm/acct/*` — output from accounting daemon (trim)
- `/usr/adm/lpd-errs` — output from `syslog` daemon (truncate)
- `/usr/adm/messages` — output from `syslog` daemon (truncate)
- `/usr/adm/sulog` — record of logs, moved to `OLD*` at reboot (trim)
- `/usr/lib/cronlog` — record of `cron` activity, moved to `OLD*` at reboot (trim)
- `/usr/mail/*` — undelivered mail (remove)

- `/usr/preserve/*` — files from aborted `ex/vi` sessions (remove)
- `/usr/spool/lp/log` — output from `lp` (truncate)
- `/usr/spool/lp/oldlog` — output from `lp` (truncate)
- `/usr/spool/mqueue/syslog` — output from `syslogd` (trim)
- `/usr/spool/uucp/LOGFILE` — record of UUCP transactions (trim)

3.2 Serving read-only files via NFS

The manual pages, font files, and even binary executables can be served over NFS, saving substantial amounts of disk space. One danger of this approach, however, is that several client machines can be adversely affected by a problem on a single server. In addition, the interactive response time of the server may be increased by the duties it performs for other machines.

In any case, before you set up a directory for shared access, you need to resolve several questions:

- Does the directory contain only shared files?
- Does the directory pose any unusual security considerations?
- Can a client system boot without access to the directory?
- Will client system responsiveness be adversely affected by network delays in accessing the files involved?
- Is the server system expected to be continuously available?
- Is use of the directory so frequent as to constitute an unreasonable drain on the server or the network itself?

Not all of these issues are absolute, and the use of multiple servers can help to share the workload, reducing the impact of a failed server. Take care, however, not to diminish the overall reliability and performance of the network unduly.

Once the decision has been made to serve a directory, the actual procedure is simple.

1. Put the directory into the client's `/etc/fstab` file.
2. Put the directory into the server's `/etc/exports` file.
3. Try the configuration to make sure that everything is working properly.
4. Back up the contents of the directory onto disks or tapes.
5. Unmount the served directory.
6. Remove all files and directories under the mount point(s) of the client(s).
7. Remount the served directory.

Caution: If the served directory contains binary executables, those commands will become unavailable after the client copies are removed. They will become available again only when the served copies are remounted. Therefore, make sure you have spare copies of any needed commands that will be affected by this procedure.

3.3 Compressing Infrequently used files

The tools available under A/UX for compressing files are `compact`, `compress`, and `pack`. These tools reduce the size of files, saving disk space. The amount of disk storage that you can gain from such compression can be substantial, but it takes longer and is relatively more difficult to access the compressed versions than to access noncompressed files.

Thus, only large and infrequently used data files are suitable candidates for compression. For example, in A/UX, the manual page files are shipped in compressed form, saving several megabytes of storage. The `man` command has been specially written to deal with compressed files; however, the execution time of the command is relatively longer than that of other commands.

All of these compression tools reduce the size of text files by about 50%. Their performance on binary files is poorer, however. The

`compress` utility saves about 40%, and the other two programs save only about 25% of the original storage. Consequently, `compress` should be used when binary or mixed files are involved.

There are also differences in the time the utilities take to run. On text files, `pack` is slightly faster than `compress`. On binary files, however, `compress` is slightly faster. In both cases, however, `compact` takes about ten times as long as the faster of the other two. Clearly, `compact` should be used only when compatibility with another machine requires it.

3.4 Usage notes

All three programs remove their input files during the compression process. The companion decompression programs (`uncompact`, `uncompress`, and `unpack`) do the same. The user should therefore make a copy before compression (or decompression), if the original version is to be retained.

Alternatively, the appropriate "snapshot" decompression program (`ccat`, `zcat`, or `pcat`, respectively) can be used. These tools are analogous to the `cat` command in displaying the uncompressed version of the data on the screen, while leaving the compressed file in place. The `man` command thus uses `pcat` for compressed manual pages, and `cat` for uncompressed ones.

3.4.1 Extensions to names of compressed files

A word on file naming is in order at this point. The compression programs append a two-character suffix to the names of their output files. For example, the `compress` utility converts the file `sample` into the compressed file `sample.Z`.

This renaming can occasionally cause unexpected and even dangerous results, however. If the input file name exceeds 12 characters, the added characters will cause the new name to exceed 14 characters. A/UX will truncate any characters past the first 14, possibly overwriting the input file.

3.4.2 Compressing an archive of files

You may sometimes need to compress a collection of small files. You can achieve a substantial savings in storage by first producing a `cpio` or `tar` archive, then compressing the archive. Each A/UX file contains an average of half a block of dead storage, because only whole

blocks are allocated by the file system. The archive utilities eliminate this waste, saving considerable storage space. Note, however, that the same archiving utility will be needed during the decompression process.

3.5 Automating system administration with `cron`

A variety of A/UX system administration tasks need to be done on a periodic basis. Some of these tasks, such as backups onto removable media, may require manual intervention. Most, however, can be run automatically by the A/UX `cron` utility.

There are several benefits of using `cron`. For example, `cron` cannot forget to perform its functions, as a human might, it never goes on vacation or takes sick leave, and it is quite willing to do things at awkward times, such as 2 A.M. Also, `cron` can do things very frequently; as often as once a minute.

Once an activity has been chosen for automation via `cron`, the system administrator must decide who is to perform the action. The `cron` utility accords a command the same privileges of the user running the command. Since the root user has so few restrictions, `cron` should typically be run from some less powerful account.

1. The administrator must ensure that the chosen account is able to use `cron`. The file `/usr/lib/cron/cron.allow` lists the names of the users allowed to run `cron`. Users can be either personal accounts, such as `joe`, or administrative logins, such as `lp`. You enter the names of the users into this file to give them access to `cron`. If this file is empty or absent because you've deleted it, then the system checks a file called `cron.deny`. This file lists users who are denied access to `cron`, and again, you enter names into the file. It is your choice whether to use `cron.allow` or `cron.deny` to control access to `cron`. There is little reason to use both. If neither file exists, only the root user is allowed to use `cron`.

In general, using the `cron.allow` file is more secure than using `cron.deny`, since the administrator must actively approve each user who might wish to use `cron`. It is more convenient, however, to use `cron.deny`, and an administrator who feels a site's security needs are low might choose this

option. Lack of both files is seldom appropriate, since all `cron` commands must then be invoked as if by the root account.

2. Now test the desired command, using a fresh login session. Give the substitute user command (`su`) for the account that `cron` will emulate.

```
su lp
```

3. Attempt to execute the desired command just as it will appear in the user's `cron` control file. This will ensure that the permissions, command format, and user environment will be the same as that used by `cron`. (Collect multiple commands into single shell scripts, if they need to be run at the same time.)

```
nightly lp1 lp2 lp3
```

4. Use the `crontab(1)` command to install the new command into the `crontab` file. Use `crontab -l > snapshot` to get a copy of the current file.

```
0 2 * * 0 weekly lp1 lp2 lp3
```

5. Use `crontab snapshot` to edit the resulting file and then the system copy.

```
0 2 * * 1-6 nightly lp1 lp2 lp3
0 2 * * 0 weekly lp1 lp2 lp3
```

Monitor the results of a few executions of the automated activity to make sure the system is working smoothly.

4. AppleCD-ROM and A/UX

A/UX supports CD-ROMs on the AppleCD SC®, if they contain A/UX file systems. This lets A/UX users take advantage of the large storage capacity of CD-ROM. A CD-ROM might contain a large number of database files, source code files, or documentation.

You can use a CD-ROM already containing an A/UX file system, or you can put an A/UX file system on a CD-ROM. Putting an A/UX file system on a CD-ROM requires copying certain information onto it. See *AppleCD SC Developer's Guide* for more information on creating information for a CD-ROM.

Note that A/UX does not provide support of audio CD-ROMs or CD-ROMs that use the High Sierra format. (The Macintosh OS does provide support for audio CD-ROMs and CD-ROM discs that use the High Sierra format.)

Follow these steps to install an AppleCD SC:

1. Install your AppleCD SC according to the directions in *AppleCD SC Owner's Guide*.
2. Bring your system up in single-user or multi-user mode.

You can now access your CD-ROM just as you access a standard hard disk.

4.1 Mounting a CD-ROM as an A/UX file system

If you have a CD-ROM that contains an A/UX file system, you can mount the CD-ROM just like any other read-only file system. For example, if your AppleCD SC has SCSI ID 4, then you can insert a CD-ROM containing an A/UX file system into the drive and enter

```
mount /dev/dsk/c4d0s0 /cdrom
```

You can read files on a CD-ROM just as you do those on any other disk. After mounting the CD-ROM, for example, enter

```
ls -R /cdrom
```

and you might see a listing like this:

```
./mammals:
    cats      lions
    dogs      mice
    elephants tigers
    giraffes  squirrels
    hippos    zebras

./programs
    prog1.c      prog6.c
    prog2.c      prog7.c
    prog3.c      prog8.c
    prog4.c      prog9.c
    prog5.c      proga.c
```

4.2 Mounting remotely

You might want to do a remote mount of the file system on the CD-ROM, making this file system available to everyone on your network. (You must have an NFS or NFS with Yellow Pages kernel to mount a file system remotely. See *A/UX Network System Administration* for information on setting up a network.) To allow others to do a remote mount of the A/UX file system that is on the AppleCD SC, create a mount directory for the file system, and modify `/etc/fstab` and `/etc/exports`. Create a mount directory for the CD-ROM, giving it a name that will be meaningful to users on other machines, for example,

```
mkdir /cdrom
```

To mount the file system automatically when you enter multi-user mode, add a line for the new file system to `/etc/fstab` on the NFS server, for example,

```
/dev/dsk/c0d0s0 /      ignore      rw      1 1
/dev/dsk/c4d0s0 /cdrom  5.2      ro       2 2
```

The second line specifies the device with SCSI ID 4 as an A/UX file system that will be mounted under `/cdrom`.

Modify your `/etc/exports` file to export this file system, for example,

```
/cdrom          #export to everyone on the network
```

Check that the file system is available to others by giving the command

```
showmount -e
```

You should see a line similar to

```
export list for hostname1:
/cdrom          everyone
```

To mount the file system from another machine, put an entry in `/etc/fstab` for this file system or use the `mount` command. For example, to mount the file system remotely from an NFS client machine using the `mount` command, enter

```
mount -o nfs,soft,ro hostname1:/cdrom /mnt
```

After remotely mounting the file system, you can access and read the files just as you would the files on any hard disk.

BLANK PAGE

PLACE TAB
HERE

OTHER
PERIPHERAL
DEVICES

BACK OF TAB

Chapter 7

Managing Other Peripheral Devices

Contents

1. Introduction	1
1.1 Ports	2
2. Using <code>autoconfig</code> to add a device requiring kernel modification	3
2.1 Using <code>/etc/newunix</code>	4
2.2 Adding new devices	5
3. Setting up a terminal	5
3.1 The <code>/etc/inittab</code> file	5
3.2 The <code>/etc/gettydefs</code> file	7
3.3 Using another computer as a terminal	8
3.4 Attaching a Macintosh Plus as a terminal	9
4. Setting up a modem	12
4.1 Dial-out access	12
4.2 Dial-in access	15
4.3 The modem as a dial-in and a dial-out device	16
4.4 Setting up an Apple personal modem	17
5. Setting up a printer	18
5.1 Definitions	19
5.2 <code>lp</code> commands	20
5.2.1 Commands for general use	21
5.2.2 Commands for <code>lp</code> administrators	21
5.3 Determining <code>lp</code> system status	22
5.4 The <code>lp</code> scheduler	23
5.4.1 Installing the scheduler	23
5.4.2 Stopping and starting the scheduler	23
5.5 Configuring the <code>lp</code> system	25
5.5.1 Introducing new destinations	26
5.5.2 Modifying existing destinations	28

5.5.3	Altering the system default destination	29
5.5.4	Removing destinations	30
5.6	Setting up an Imagewriter or Laserwriter printer	31
5.6.1	Adding an ImageWriter II printer	31
5.6.2	Adding a LaserWriter printer	33
5.6.3	If you are connecting a LaserWriter II NT or NTX	36
5.6.4	Changing from a LaserWriter to an ImageWriter	37
5.6.5	Setting up other printers	39
5.7	Writing printer interface programs	41
5.8	Using the lp system	43
5.8.1	Allowing and refusing requests	45
5.8.2	Allowing and inhibiting printing	46
5.8.3	Moving requests between destinations	47
5.8.4	Canceling requests	48
5.9	lp system troubleshooting	48
5.9.1	Problems starting lpsched	48
5.9.2	Restarting lpsched	49
5.9.3	Repairing a damaged outputq file	50
5.9.4	lp system files	50
5.9.5	lp system command permissions	52

Chapter 7

Managing Other Peripheral Devices

1. Introduction

This chapter discusses management of all peripheral devices except hard disks, which are discussed in the previous chapter. Managing peripheral devices involves connecting devices such as terminals, modems, and printers to your computer; maintaining them; and, when necessary, disconnecting them. This chapter helps make these tasks as painless as possible.

Besides this introduction, this chapter consists of four main sections. Section 2 discusses the universal procedure to connect a device requiring kernel modification. Sections 3 and 4 discuss how to connect a terminal and a modem. Section 5 describes how to connect a printer and administer its spooler program, which holds print jobs when the printer is busy.

Peripheral devices that you connect to your machine fall into two categories: those that require adding a device driver to the A/UX kernel and those that don't. A device driver is the software interface between A/UX and the peripheral device. All A/UX peripheral devices require a device driver. The reason for the two categories is that the A/UX kernel has the drivers already built in for commonly used Macintosh peripherals, whereas you must add device drivers for other, less commonly used or third-party devices. It is not readily apparent into which category a particular device falls. In general, if adding the device involves adding an expansion card (such as an Ethernet® card) to the computer, you also need to add a device driver.

You might want to connect another computer to your A/UX system. You can do this in at least two different ways. You can connect both computers to a standard network, such as Ethernet. This is known as *networking*, and it is explained in *A/UX Network System Administration*. Alternatively, you can connect a computer through a serial port to the Macintosh II, and it will act as a peripheral device of

the Macintosh II. This method is discussed here under "Setting Up a Terminal."

To understand some sections in this chapter, you need to be familiar with the file `/etc/inittab`. This file is discussed in Chapter 2, "System Startup and Shutdown" and `inittab(4)` in *A/UX Programmer's Reference*. See also "Changing Run Levels: `init`" in Chapter 2, because incautious use of the `init` command may affect other users.

1.1 Ports

It is important that you understand the concept of a port when you are working with peripheral devices. A computer communicates with other equipment through a port. You can attach peripheral devices such as printers and additional terminals by connecting them (via cables or connectors) to the ports.

There are two aspects to a peripheral connection: the hardware that connects the device to the computer and the software that enables the two to communicate. This chapter concentrates primarily on the software. For the hardware aspect of connecting a device, refer to the manual that comes with the hardware.

For now, look at the back of your computer and find the two serial ports. They are round and about half an inch in diameter. Each port has eight holes. The port identified by the phone icon on the back of the Macintosh II computer has the A/UX device name `/dev/tty0`; it is also called `/dev/modem`. The port identified by the printer icon has the A/UX device name `/dev/tty1` and is also called `/dev/printer`. You will be connecting your devices to these serial ports. (Note that these ports can be switched if desired; you can attach a modem to port `tty1` and a printer to port `tty0`.)

In addition to physically connecting a device to the computer, which in most cases is a rather simple operation, you must let the computer know what type of device is attached to which port and what the computer must do in each case. This is more involved but not difficult.

2. Using `autoconfig` to add a device requiring kernel modification

One feature unique to A/UX is the `autoconfig` program. This program serves two purposes. When A/UX is booted, `autoconfig` automatically checks for consistency between the cards in the expansion slots and the software in the kernel. If a discrepancy exists between the two, `autoconfig` corrects it and reboots the system. This feature of `autoconfig` is explained fully in Chapter 2 of this book.

The second purpose of `autoconfig` is sparing you the tedium and potential pitfalls of adding a new peripheral device. The `autoconfig` program configures the device driver and any software modules into the kernel for you.

Note: You do not need to use `autoconfig` to add terminals, printers, or modems to the A/UX system. Support for these devices is already built into the standard A/UX kernel. You do need it to add an Apple Tape Backup 40SC to the A/UX system.

1. To add a new software module or driver to the kernel, first run the script `/etc/newunix`. The `/etc/newunix` program installs the files `autoconfig` needs to add a software module to the kernel. You specify the device driver or software module you want to add to the kernel as a parameter to `/etc/newunix`. For example, the command

```
/etc/newunix tc
```

installs the device driver for the Apple Tape Backup 40SC.
2. After running `/etc/newunix`, use `autoconfig` to create the new kernel containing the new software modules. For example, the command

```
autoconfig -v -u -I -S /etc/startup
```

gives a verbose account of the actions performed by

autoconfig, places a list of startup programs to run at boot time in the /etc/startup file, and creates a new kernel in /unix.

3. Now you must reboot the new kernel before using the newly added device or the new software module. Shut down the system and reboot the new kernel by following the directions in Chapter 2 of this book.

2.1 Using /etc/newunix

You use /etc/newunix to install the files for a particular software module into the directories required by autoconfig. By default, autoconfig builds a new kernel based on the information in /newunix, the object modules in /etc/boot.d, and the files in /etc/master.d. Once you have added a new software module or driver to your system, it is always included when autoconfig builds a new kernel, unless you explicitly remove it.

- ☐ To see which modules are currently configured in the kernel, enter the command

```
ls /etc/master.d
```

If you see a result similar to

```
BNET ae6 nfs slots toolbox tc
```

then the current kernel is configured for B-NET (BNET) and NFS (nfs), for an EtherTalk® card (ae6), for slots support (slots), A/UX toolbox support (toolbox), and for Apple Tape Backup 40SC support (tc).

- ☐ You can also use /etc/newunix to remove software modules from the kernel. For example, to remove the software driver for the Apple Tape Backup 40SC, enter

```
/etc/newunix notc
```

Now enter

```
ls /etc/master.d
```

You should see a result similar to

`BNET ae6 nfs slots toolbox`

Note that the `tc` object module has been removed from the `/etc/boot.d` directory. Run

`autoconfig -v -S /etc/startup`

to create a new kernel that does not include the `tc` driver. The `autoconfig` program builds a new kernel based on the object modules in `/etc/boot.d`. Because the `tc` module is not in this directory, the `tc` module is not included in the new kernel. Shut down your system and reboot to begin using the new kernel.

2.2 Adding new devices

When you add a new device, you must run the installation scripts provided with that device. An installation script typically installs files for that device in `/etc/install.d`. Follow the instructions provided with your device to install any software modules into the kernel. Typically, after you run the installation script for the new device, you will need to run `/etc/newunix`, then `autoconfig`, and then reboot your system, as described in the previous section.

3. Setting up a terminal

When you set up your computer and start running A/UX, you'll have at least one terminal working. This first terminal is referred to as the console. You can add additional terminals as you need them.

Before the computer can communicate with a peripheral device, it has to know what is expected of it at each port. To do this, the computer uses the system initialization instructions found in the `/etc/inittab` file and the TTY definitions (`gettydefs`) found in the `/etc/gettydefs` file.

The `/etc/gettydefs` file contains information used by the `/etc/getty` program (the process that waits for a user to log in) to determine settings of a `getty` at a given port. The login prompt for each port is also set in this file.

3.1 The `/etc/inittab` file

The first step in telling the system about a new terminal is to change the `/etc/inittab` file. The `init` program uses this file to decide which processes to run when the system comes up; see Chapter 2,

"System Startup and Shutdown." *It is a good idea to make a copy of the /etc/inittab file before you make any changes.* This provides protection against errors when you're modifying the file.

1. Change to the /etc directory and enter

```
cp inittab inittab.old
```

2. Now open the /etc/inittab file using a text editor (for example, vi). Three of the lines in the file should look like these, though not necessarily in this order:

```
co::respawn:/etc/getty console co_9600
00:2:respawn:/etc/getty tty0 at_9600
01:2:respawn:/etc/getty tty1 at_1200
```

These lines are each followed by comments documenting what the entry represents and (if applicable) how to enable it. These comments, respectively, are

```
# Console port
# Port tty0 (modem); set to "respawn"
# Port tty1 (print); set to "respawn"
```

Each of these entries is for a given port. The entries are divided into fields, separated by colons, with the form

id:run-level:action:command

The following is a typical entry:

```
00:2:respawn:/etc/getty tty0 at_9600 # port tty0
```

In this case, /etc/getty accepts two arguments. The first, tty0, specifies the port on which the command should run. The second, at_9600, is a label. It tells init to read the /etc/gettydefs file and use the information contained in the entry beginning with at_9600 to set up communications with port tty0.

The *action* field in /etc/inittab entries for terminals should be respawn. If there is anything other than respawn in the *action* field and you want to allow people to log in at that terminal, correct the *action* field and close the file.

3.2 The /etc/gettydefs file

Another file that is very important in peripheral management is /etc/gettydefs. This file is composed of individual entries, each with five fields separated by a number sign (#). *Note that each entry is separated from the others by a blank line.* Except for the *prompt* field, you may insert whitespace (blanks or tabs) between the fields for readability. The entries are of the form

label # initial-flags # final-flags # flow-control # prompt # next-label

where the fields are interpreted as follows:

label

String that *getty* tries to match so that it can use the entry. If the second argument to /etc/getty in an /etc/inittab entry is, for example, co_9600, the entry that starts with this string is used.

initial-flags

Used to decide how the terminal is set up before log in. The only critical flag at this point is the B flag, which is used to decide the communications baud (speed). In the example below, the flag is set to 9600, but it could be any valid baud.

final-flags

Take effect when login is executed. Again, speed (for instance, B9600) is critical. SANE is a composite setting; it takes care of other important terminal settings without your having to set them individually. TAB3 specifies that tabs will be sent to the terminal as spaces. HUPCL specifies that the line should hang up on closing the connection. This is generally set for terminals that use a phone line through a modem. To subtract a particular attribute from a setting, prefix it with a tilde (~). Thus, SANE ~PARENB sets the terminal to have all the attributes of SANE with the exception of PARENB (parity). For more information on these flags, see *termio(7)* in *A/UX System Administrator's Reference* and *gettydefs(4)* in *A/UX Programmer's Reference*.

flow-control

Specifies the type of flow control to be used on the line. The settings can be APPLE, DTR, MODEM, and FLOW. Again, you

can subtract a setting by prefixing it with a tilde (~).

prompt

Login prompt that will appear at the terminals.

next-label

Label for getty to try in case the current entry causes a failure. If getty cannot read the keyboard input using the current definitions, it tries the gettydef specified in this field. For instance, if the user logs in at a terminal set up to communicate at 4800 baud but the getty being sent to that terminal specifies 9600 baud, the getty would not accept the input. When this happens, getty looks at this field for an alternative setting. It tries entries in sequence until it finds one expecting input at 4800 baud.

Enter

```
more /etc/gettydefs
```

Two of the entries that scroll on the screen after you use this command should look something like the following, although the whole file may be several pages long.

```
co_9600# B9600 # B9600 SANE TAB3 # ^MODEM ^DTR ^FLOW  
# \r\n\nApple Computer Inc. A/UX\r\n\nlogin: # co_4800
```

```
tt_9600# B9600 # B9600 SANE TAB3 ^MODEM ^DTR ^FLOW  
# \r\n\nApple Computer Inc. A/UX\r\n\nlogin: # tt_4800
```

Note: In the this example, output lines have been wrapped onto two lines. When a line in the file has more characters than will fit on a single terminal line, the line will wrap onto the next screen line even though there is only one corresponding file line.

3.3 Using another computer as a terminal

It is possible to connect another computer to an A/UX system just as a terminal is attached, through a serial line. For example, you can treat a Macintosh computer running MacTerminal® exactly like a terminal. As long as the files /etc/inittab and /etc/gettydefs are configured to allow logins on the appropriate port, successful communication can take place, even though the A/UX system has no

way of knowing that it is communicating with a computer and not merely a terminal.

There are numerous advantages to replacing a terminal with a personal computer that emulates a terminal. Some of these advantages include the abilities to scroll back through output and to transfer files between the computers.

3.4 Attaching a Macintosh Plus as a terminal

Attaching a terminal to your Macintosh II allows a second user to access A/UX while you or someone else is logged in at the console. This section describes how to attach a Macintosh Plus, running a terminal-emulator application such as MacTerminal, to your Macintosh II. (These instructions apply to a Macintosh SE as well as a Macintosh Plus.)

To connect a Macintosh Plus as a terminal, you need an Apple system cable, such as a Macintosh Plus-to-ImageWriter® II cable (part 590-0552 or M0187), with mini-8 connectors at both ends.

1. **Configure your terminal-emulator application on the Macintosh Plus.**

Start your terminal-emulator application on the Macintosh Plus. Consult the user's guide for your application to set the terminal characteristics shown below. (If you are using MacTerminal, for example, you configure the application by selecting Terminal and Compatibility from the Settings menu.)

- **terminal type:** VT100 (The VT100, a popular terminal manufactured by Digital Equipment Corporation, is emulated by nearly all communications programs. The VT100 is the terminal A/UX expects by default to find at `/dev/tty0`. See `ttytype(4)` in *A/UX Programmer's Reference* for more information about how A/UX is configured for default terminals.)
- **line width:** 80 columns
- **mode:** ANSI
- **baud:** 9600
- **bits per character:** 8
- **parity:** none
- **connection port:** modem

- connection: to another computer (that is, instead of to a modem)
 - handshake: XON/XOFF
2. Plug one end of the cable's circular connector into the modem port on the back of the Macintosh II. The modem port is identified by the symbol of a telephone handset.
 3. Connect the free end of the cable to the modem port on the Macintosh Plus. The modem port is located on the back of the Macintosh Plus and is identified by the same symbol as the modem port on the Macintosh II.
 4. Bring A/UX up to multi-user mode if it isn't already. If the login: prompt appears on your console, then your system is already in multi-user mode. If your system is in single-user mode, or if the root user's command prompt appears on your console and you're not sure if the system's in multi-user mode, enter the command

```
init 2
```

This ensures that A/UX is in multi-user mode. If you're prompted to check the file systems, enter n.

5. From the Macintosh II console, log in to A/UX as the root user.

If the root user's command prompt appears on the console, you're already logged in as the root user.

However, if the login: prompt appears instead, enter root. Provide your password when prompted and press RETURN to accept the mac2 terminal type. You need root privileges to make the system configurations explained in later steps.

6. Make a copy of /etc/inittab.

When you receive the root user's command prompt after finishing step 5, type

```
cp /etc/inittab /etc/inittab.old
```

and press RETURN. When you change an important system file like this, it is always a good idea to save a copy in case you make a mistake. You can then copy the old file back over the changed version and return your system to its previous state.

Note: After entering this command, you should immediately see the root user's command prompt again. If the system returned any additional messages, you've entered the command incorrectly—in this case, reenter the command more carefully.

7. Use your favorite text editor to edit `/etc/inittab`. Change the line for `tty0` to replace `off` with `respawn` so that the beginning of the line looks like this:

```
00:2:respawn:/etc/getty tty0
```

8. At the command line prompt enter `initq` to effect the changes in `/etc/inittab`.

Type

```
init q
```

and press RETURN.

9. Verify that the `getty` process is running.

Type

```
ps -ef | grep getty
```

and press RETURN.

A line similar to the one below should appear on your screen:

```
root  82  1  0 11:43:37  0 0:01 /etc/getty tty0
at_9600
```

The numbers in your output will be different, but a 0 should appear after `tty`, showing that a `getty` process has successfully spawned at `/dev/tty0` (the modem port). This is the process that will serve the terminal.

Note: Do you see different output? If you receive no output from the command, or if a number other than 0 appears in the third column, enter the following command:

```
cp /etc/inittab.old /etc/inittab
```

After entering this command, begin these instructions over again at step 7.

10. Log in to A/UX from the Macintosh Plus.

You should now have a login prompt on the Macintosh Plus. Log in as the root user or any other valid user to test the connection. Enter your password when prompted, and press RETURN to accept the VT100 terminal type. Your Macintosh Plus is now serving as a functioning terminal for A/UX.

Note: You don't get a login prompt? If you don't get a login prompt on the Macintosh Plus, check the cable connection or your settings on the terminal-emulation application. If the command in step 9 was successful, then the problem is not likely to be related to A/UX.

4. Setting up a modem

A modem can function in incoming or outgoing mode. On the A/UX system, however, it cannot do both at one time. You can switch between these modes using multiple run levels; see `init(1M)` in *A/UX System Administrator's Reference* and Chapter 2, "System Startup and Shutdown." Or, you can set up a dial-out modem on one port and a dial-in modem on the other port. This section uses the examples of a terminal on port `tty0` and a modem on port `tty1`. This is arbitrary. If you want, you can have a modem on port `tty1` and a printer on port `tty0`, even though the hardware ports have a modem icon at `/dev/tty0` and a printer icon at `/dev/tty1`.

4.1 Dial-out access

Suppose `/etc/inittab` file has the following entries

```
co::respawn:/etc/getty console co_9600 # Console port
00:2:respawn:/etc/getty tty0 at_9600   # Port tty0
01:2:respawn:/etc/getty tty1 at_9600   # Port tty1
```

and you want the modem on port `tty0` to work as an outgoing device.

1. Find the line

```
00:2:respawn:/etc/getty tty0 at_1200 # Port tty0
```

and change it to

```
00:2:off:/etc/getty tty0 at_1200 # Port tty0
```

You can also change it to

```
00:2:off:/etc/getty tty0 mo_1200 #Port tty0 dialout
```

making sure there is a `mo_1200` entry in `/etc/gettydefs`. This change is optional, but it makes the `/etc/inittab` file easier to read.

Because the third field contains `off`, the line now says that, at run level 2, no `getty` is present on port `tty0`. (See Chapter 2, "System Startup and Shutdown," for a discussion of run levels.) In this condition, the modem functions as an outgoing device only.

2. You have to modify one more file. Change to the directory

```
/usr/lib/uucp
```

Most computer communications programs are located in this directory.

3. Make a copy of the file `L-devices` and tuck it away somewhere for safety's sake.
4. Now open the file with a text editor and add the following lines at the bottom:

```
DIR tty0 tty0 1200
DIR tty0 tty0 300
```

These two lines tell the system that port `tty0` is connected physically via a cable to another computer and that this cable can be used for transmission speeds of either 300 or 1200 baud. If 1200 baud is not the speed you want (either because you want to transmit at 300 baud or because the other computer receives at 300 baud), you can select 300 baud. For more information about the `L-devices` file, see `cu(1C)` and `uucp(1C)` in *A/UX Command Reference* and Appendix A, "The UUCP System."

5. Now that you've made the necessary preparations, you're probably eager to use your modem. To enable outgoing calls, reboot your system (see "Using the shutdown Program" in Chapter 2). When the system is in multi-user mode, the new entry in `/etc/inittab` will take effect, and your modem will function in outgoing mode.
6. To make a call, you need to know the phone number of a modem attached to another computer set up to receive calls. Once you have a number, you can use the `cu` command (among others) to get your modem to talk to other computers. This is the command used in this manual for testing purposes.

Most modems operate at 300 or 1200 baud (or higher). Using a modem at 300 baud can be extremely slow. If you can, operate the modem at 1200 baud. You can do this using the `cu` flag option `-s`:

```
cu -s[speed]
```

In this case,

```
cu -s1200
```

Another key to getting `cu` to work is the `-l` option. The `-l` option stands for "line" and tells `cu` which port the modem is on. The command line that allows you to dial out is

```
cu -s1200 -ltty0
```

Type this line and press RETURN. The word Connected should appear. This means you are connected to the modem. If you have any problem at this stage, or later, check that the ownership of `/dev/tty0` is the same as the ownership of `/usr/bin/cu`.

7. The next sequence works only if you are using a Hayes-compatible modem, but the principle is the same in all cases: Once you are connected to the modem, there is a specific way of communicating with it and making it do what you want. If your modem is not Hayes compatible, your modem owner's manual will have information on how to communicate with it.

Type the command that tells your modem to dial a phone number. If your modem is Hayes compatible, try this:

ATDT *phone-number*

The *phone-number* field must be the number of a computer ready to receive a call. If you are in an office and must request an external line by dialing a number, say 9, before the phone number, then the command to try is

ATDT 9,*phone-number*

This tells the modem to dial 9, wait, and then dial the phone number.

8. Once you are connected to the other computer, you will see that computer's login prompt appear on your screen. You can now log in and treat this remote computer as if you were sitting at a terminal in front of it. See *cu(1C)* in *A/UX Command Reference*.
9. When you are ready to break the connection with the other computer, you must log out.
10. Once you are logged out, wait one second, then type +++ and wait another second. Your modem answers OK on your screen.
11. Enter ATH. This tells your modem to hang up. Again, if your modem is not Hayes compatible, the manual that comes with it describes how to disconnect from a remote computer. The modem responds with OK.
12. Type the two-character sequence "tilde-dot":

~.
This terminates the session with cu. The word *Disconnected* appears, and you are back at your own computer.

4.2 Dial-In access

It's often to your advantage to allow other people or other computers to use your system via dial-in access. Once your A/UX system is set up to receive calls, anyone dialing your system can use it as if he or she were connected directly to your computer via a terminal.

1. To make your modem work as an incoming device, find this line in `/etc/inittab`

```
00:2:off:/etc/getty tty0 mo_1200 # Port tty0 dialout
```

and change it to

```
00:2:respawn:/etc/getty tty0 mo_1200 # Port tty0 dialup
```

This line now tells the system to expect a `getty` on port `tty0` at run level 2. This allows the modem on port `tty0` to function as an incoming device only.

2. To enable incoming calls, reboot your system (see "Using the shutdown Program" in Chapter 2). When the system is in multi-user mode, the new entry in `/etc/inittab` will take effect and your modem will function in incoming mode. You might also have to instruct your modem to auto-answer; consult the modem manual for details.

4.3 The modem as a dial-in and a dial-out device

Under A/UX, the modem cannot operate as a dial-in device and a dial-out device at the same time. The presence of a `getty` prevents dial-out access. Similarly, the absence of a `getty` prevents dial-in access.

Using multiple run levels can make switching the modem from a dial-out device to a dial-in device and back again an easy process. A few changes are required for this.

1. To initiate these changes, use a text editor to edit the file `/etc/inittab`. Find the line referring to `tty0`. Change it to

```
do:2:off:/etc/getty tty0 mo_1200 # Port tty0 dialout
```

Note that you should change the *id* field on this entry before you add the next line.

2. Add the following line, which also refers to `tty0`:

```
du:3:respawn:/etc/getty tty0 mo_1200 # Port tty0 dialup
```

These two lines mean that at run level 2 there will be no `getty` at port `tty0` and that there will be a `getty` at run level 3.

Therefore, at run level 2 the modem attached to port `tty0` can dial out, and at run level 3 it can be called up. If the *id* field is

not unique for each entry, you will see an error message when you change to run level 3.

3. Now you can enter from the shell

```
init 2
```

to enable dial-out mode. Conversely, you can enter the command

```
init 3
```

to enable dial-in mode. Once again, you may need to instruct the modem to enter auto-answer mode.

4. When you set up things this way in `/etc/inittab`, you must also make sure that all other processes set up to run at run level 2 continue running at run level 3. For instance, if your printer is set up to run at run level 2, you probably also want it to run when your modem is in dial-in mode. For example, if your printer is connected to port `tty1`, change the printer entry in `/etc/inittab` to

```
01:23:off:/etc/getty tty1 at_9600 # Port tty1 (print)
```

Notice the entry in the *run-level* field. It is now 23, to signify that the entry applies to run levels 2 and 3.

At this point, some of the entries in `/etc/inittab` should look like this:

```
co::respawn:/etc/getty console co_9600 # Console port
do:2:off:/etc/getty tty0 mo_1200      # Port tty0 dialout
du:3:respawn:/etc/getty tty0 mo_1200  # Port tty0 dialup
01:23:off:/etc/getty tty1 at_9600    # Port tty1 (print)
```

4.4 Setting up an Apple personal modem

Setting up an Apple Personal Modem requires setting the baud rate and running a program that sets the modem for auto-answer dial-in. Both modifications are accomplished with changes to `/etc/inittab`. To set the Apple Personal Modem for auto-answer and dial-in, and set the baud rate, change the `/etc/inittab` entry from

```
00:2:off:/etc/getty tty0 at_9600
```

to

```
00:2:respawn/etc/apm_getty tty0 mo_1200
```

For dial-in lines, `/etc/apm_getty` needs to run instead of `/etc/getty`. The `/etc/apm_getty` program sends the control sequences that enable auto-answer to the modem and then executes the `/etc/getty`.

5. Setting up a printer

A variety of printers can be used with A/UX. The exact steps you take when setting up an Apple ImageWriter® or LaserWriter® printer are given in this section. In addition, the section covers procedures and concerns related to setting up any printer with A/UX, although you will need to supplement that information with details from the owner's guide accompanying your printer.

Note: A/UX supports the LaserWriter II NT and the LaserWriter II NTX but not the LaserWriter II SC.

You can also print on a printer shared with other computers by using the AppleTalk® network. For further information, see the *A/UX Network System Administration Guide*.

This section also introduces the commands necessary to use the `lp` system. The `lp` system is a program that solves the first problem you run into when administering a printer on a system with more than one user: how to prevent two people from using one printer at the same time. The `lp` system is a collection of programs and files used to manage a printer's operations. (The historical acronym "`lp`" stands for "line printer.")

Note: If you or your users are accustomed to BSD systems, you may be pleased to know you can continue using the `lpr` command to use the `lp` system; the A/UX version of `lpr` is a shell script designed to emulate the functionality of the BSD `lpr`.

The `lp` system is composed of two subsystems: the spooler and the administrative system. The spooler intercepts print requests, schedules them for printing on a specified printer, or class of printers, and then selects the appropriate interface for the printer. The administrative system is a series of commands you use to configure and maintain the entire `lp` system.

When configuring the spooler, the system administrator assigns each printer a unique name by which the spooler will identify the printer. After naming the printer, you may assign the printer to a particular named class of printers. A name or class of printers is also known as a destination.

The spooler maintains a list of user print requests organized by printer name and printer class. Implementing printer classes is not required but is strongly recommended if you have multiple printers.

Classes allow a user to print a job on the first available printer in the class rather than wait for a particular printer. It's usually best to organize printer classes along the lines of printer types. For example, a class named "Laser" might contain only laser printers, and a class named "Matrix" might contain only dot-matrix printers.

The `lp` system is controlled and managed through a set of commands that

- queue or cancel requests
- query the status of requests or of the `lp` system itself
- prevent or allow queuing requests to specific printers
- start or stop the `lp` system
- change the printer configuration

5.1 Definitions

This section defines the important terms used throughout this chapter.

- A request is a print job submitted to the `lp` system using the `lp` command.
- A printer is a unique name by which the `lp` system identifies a specific printer

- A class is a name used to identify a defined list of printers. Each printer may be a member of zero or more classes.
- A destination is a printer or a class. Output is normally routed to the system default destination unless the user explicitly requests a particular printer or printer class on the `lp` command line. See `lp(1)` in *A/UX Command Reference*.
- A device is a piece of hardware such as a printer or modem that can be connected to the computer through a port.
- A device file is a file in the `A/UX /dev` directory that is associated with a particular device. For example, `/dev/lw2` is the device file that represents an ImageWriter II. Each RS-232 port on the back panel of your computer is associated with one device file. When the `lp` system writes to the device file, output is sent to the port. The `lp` system maintains information necessary to associate each printer with a particular device.
- The `lp` scheduler, called `lpsched`, schedules print requests received from the `lp` command. The `lpsched` scheduler runs continuously in the background and is usually started by the `init(1M)` process when `A/UX` enters multi-user mode. See `lpsched(1M)` and `init(1M)` in *A/UX System Administrator's Reference*.
- Each printer is controlled by an interface program, which may be shared by more than one printer. Interface programs perform such tasks as setting port speed, selecting printer options, printing banners, and perhaps filtering certain characters a particular printer may not know how to handle. The `lp` system maintains the information necessary to select the proper interface for a given printer.

5.2 `lp` commands

The commands used to administer the `lp` system can be divided into two categories: those that any user can use, and those that only the `lp` administrator can use. This section gives a short description of what each command does.

5.2.1 Commands for general use

lp

Submits a print request to the lp system. The request is printed on the default system destination or optionally routed to a specified printer or printer class. A successful request prints a message on the user's terminal similar to

request id is *dest-seqno* (1 file)

where *dest* is the name of a printer or printer class and *seqno* is a number unique across the entire lp system. See lp in *A/UX Command Reference*.

cancel

Cancels requests by printer name or request ID number (*dest-seqno* supplied by lp). Specifying the printer name cancels the job currently printing. See lp(1) in *A/UX Command Reference*.

lpstat

Gives certain status information about the lp system. Also see lpstat(1) in *A/UX Command Reference*.

disable

Prevents lpsched from routing output requests to specified printers.

enable

Allows lpsched to route output requests to printers. See enable(1) in *A/UX Command Reference*.

5.2.2 Commands for lp administrators

In each lp system, a person or persons must be designated as lp administrator to perform the restricted functions listed below. The lp login (provided with the standard A/UX distribution) owns all the files and commands associated with the lp system. Either the superuser or any user logged into the A/UX system as lp qualifies as the lp administrator.

The following commands are described in more detail later in this chapter.

lpadmin

Modifies the lp configuration. Many features of this command

cannot be used when `lpsched` is running. Also see `lpadmin(1M)` in *A/UX System Administrator's Reference*.

`lpsched`

Routes user print requests to interface programs, which do the printing on devices. Also see `lpsched(1M)` in *A/UX System Administrator's Reference*.

`lpshut`

Stops a running `lpsched`. All printing activity is halted, but other `lp` commands may still be used. Also see `lpsched(1M)` in *A/UX System Administrator's Reference*.

`accept`

Allows `lp` to accept output requests for destinations. Also see `accept(1M)` in *A/UX System Administrator's Reference*.

`reject`

Prevents `lp` from accepting requests for particular destinations. Also see `reject(1M)` in *A/UX System Administrator's Reference*.

`lpmove`

Moves output requests from one destination to another. Whole destinations may be moved at one time. This command cannot be used when `lpsched` is running. Also see `lpmove(1M)` in *A/UX System Administrator's Reference*.

5.3 Determining `lp` system status

The `lpstat` command displays on the screen the status of printing requests, destinations, and the scheduler (`lpsched`). Also see `lpstat(1)` in *A/UX Command Reference*.

Common uses of the `lpstat` command are to:

- List the status of all currently printing and pending requests you have made:

`lpstat`

- List all currently printing and pending requests of all users:

`lpstat -o`

The status information for a request includes the request ID, the

login name of the user, the total number of characters to be printed, and the date and time the request was made.

- Determine whether a printer is available to print requests:

```
lpstat -r -a -p
```

Before a request can be printed, the scheduler (`lp sched`) must be running and the particular printer must be enabled and accepting requests. This command produces the necessary information for all printers. See `accept(1M)` in *A/UX System Administrator's Reference* and `enable(1)` in *A/UX Command Reference*.

5.4 The `lp` scheduler

The `lp sched` program routes the output requests (made with `lp`) through the appropriate printer interface programs to the printers. As noted previously, before a request can be printed, the scheduler (`lp sched`) must be running and the particular printer must be enabled and accepting requests.

5.4.1 Installing the scheduler

To install `lp sched`, make sure the following line in the `/etc/rc` file is not "commented out" with number signs (`#`) at the start of the line:

```
rm -f /usr/spool/lp/SCHEDLOCK
```

Also be sure that the following line appears in `/etc/inittab`:

```
lp:2:once:/usr/lib/lpsched >/dev/syscon 2>&1
```

This starts the `lp` scheduler each time the A/UX system enters run level 2.

5.4.2 Stopping and starting the scheduler

Each time the scheduler routes a request to an interface program, it records an entry in the log file, `/usr/spool/lp/log`. This entry contains the login name of the user who made the request, the request ID, the name of the printer on which the request is being printed, and the date and time that printing first started. If a request has been restarted, more than one entry in the log file may refer to the request. The scheduler also records error messages in the log file. When `lp sched` is started, it renames `/usr/spool/lp/log` as `/usr/spool/lp/oldlog` and starts a new log file.

Note: The log files can grow substantially and eventually occupy an enormous amount of disk space. You should inspect these files periodically and, if necessary, truncate them to zero length by giving the commands

```
cp /dev/null /usr/spool/lp/oldlog
cp /dev/null /usr/spool/lp/log
```

As mentioned earlier, the lp system won't perform any printing unless lpsched is running. Use the command

```
lpstat -r
```

to find the status of the lp scheduler.

The scheduler normally begins running when the init(1M) process executes the entry in the /etc/inittab file (described previously in "Installing the Scheduler") and continues to run until the A/UX system is shut down.

The scheduler operates in the /usr/spool/lp directory. When the scheduler starts running, it checks whether a file called SCHEDLOCK exists; if it does, lpsched exits immediately. Otherwise, lpsched creates SCHEDLOCK to prevent more than one scheduler from running at the same time.

Occasionally, it is necessary to shut down the scheduler to reconfigure the lp software. To stop the scheduler, give the command

```
/usr/lib/lpshut
```

This stops lpsched, removes the SCHEDLOCK file, and terminates all printing. All requests that were in the middle of printing will be reprinted in their entirety when you restart the scheduler.

To restart the lp scheduler, use the command

```
/usr/lib/lpsched
```

Shortly after you enter this command, you can use the lpstat -r command to determine whether the scheduler is running. If not, it is possible that A/UX crashed or was halted improperly, leaving SCHEDLOCK in the /usr/spool/lp directory.

Use the command

```
ls /usr/spool/lp/SCHEDLOCK
```

to determine if SCHEDLOCK exists. If it does, enter the commands

```
rm -f /usr/spool/lp/SCHEDLOCK  
/usr/lib/lpsched
```

Wait about 15 seconds and then use the `lpstat -r` command to determine if the scheduler is running.

5.5 Configuring the lp system

The lp system configuration is determined by a set of data files in the `/usr/spool/lp` directory. Some of these files are ordinary text files, and others contain binary data.

Note: Although it is possible to change the text files with a text editor, *don't!* Altering these files by hand while `lpsched` is running may cause strange spooler behavior. The `lpadmin` command is designed with these conditions in mind and will not allow certain configuration changes to take place until you terminate `lpsched`. Always use the `lpadmin` command to reconfigure the lp system.

The `lpadmin` command is used to change the content of these files. The `lpadmin` command can have one of the following forms:

```
lpadmin -pprinter [-vdevice] [options]  
lpadmin -xdest  
lpadmin -d[dest]
```

The three flag options `-p`, `-x`, and `-d` are mutually exclusive. Also, `lpadmin` will not attempt to alter the lp configuration when `lpsched` is running, except as explicitly noted in the remainder of this section.

The rest of the section is devoted to a series of examples that illustrate possible invocations of the commands in the lp system.

As you read, you should get a clear idea (perhaps in the form of a diagram you draw as you read) of which printer classes the examples establish, which printers belong to which classes, what models apply to what printers, and so on.

5.5.1 Introducing new destinations

You can add a new printer by giving the command

```
lpadmin -pprinter [-vdevice] [options]
```

where the fields are interpreted as follows:

printer Arbitrary name that must:

- contain no more than 14 characters
- contain only alphanumeric characters and underscores
- not be the name of an existing lp destination (whether a printer or a class)

device Pathname of a hard-wired printer or other file that is writable by lp, for example, /dev/printer.

options Any of the following:

-c class

Inserts the specified *printer* into the class *class*. The *class* will be created if it does not already exist.

-e printer-to-copy

Copies the interface program for *printer-to-copy* as the new interface program for *printer*.

-h

Indicates that the device associated with *printer* is hard-wired (plugged directly in to the computer). This option is always assumed, unless the **-l** option is selected.

-i interface

Establishes the program found in *interface* as the new interface program for *printer*.

-l

Indicates that the device associated with *printer* is a login terminal.

-m *model*

Selects *model* as the model interface program for *printer*. (See "Writing Printer Interface Programs," later in this chapter.)

-r *class*

Removes printer *printer* from the specified *class*. If the specified printer is the last member of the class, the class will also be removed.

-v *device*

Associate a new device *device* with the printer *printer*. The *device* must be a pathname of a file that is writable by *lp*.

When adding a new printer to the *lp* system, you must select the printer interface program. You may specify this in one of three ways:

- You may select it from a list of model interfaces supplied with *lp* in the `/usr/spool/lp/model` directory (**-m *model***).
- It may be the same interface that an existing printer uses (**-e *printer-to-copy***).
- It may be a program supplied by the *lp* administrator (**-i *interface***).

You may add the new printer to an existing class or to a new class (**-c *class***). New class names must conform to the rules that govern new printer names.

Here are some examples of how printers might be named:

- Create a printer called *pr1* whose device is `/dev/printer` and whose interface program is the model *hp* interface:

```
/usr/lib/lpadmin -ppr1 -v/dev/printer -mhp
```

- Add a printer called *pr2* whose device is `/dev/tty1` and whose interface is a variation of the model *prx* interface:

```
cp /usr/spool/lp/model/prx newint
```

edit newint and introduce variations

```
/usr/lib/lpadmin -ppr2 -v/dev/tty1 -inewint
```

- Create a printer called `pr3` whose device is `/dev/tty1`. Printer `pr3` will be added to a new class called `c11` and will use the same interface as printer `pr2`:

```
/usr/lib/lpadmin -ppr3 -v/dev/tty1 -epr2 -cc11
```

5.5.2 Modifying existing destinations

You can use `lpadmin` to modify existing destinations. Always make modifications with respect to a printer name (`-pprinter`). The form of the command is

```
lpadmin -pprinter options
```

These are the options available for modifying existing destinations:

- c class**
Add the printer to a new or existing class.
- e printer-to-copy**
Change the printer interface program.
- i interface**
Change the printer interface program.
- m model**
Change the printer interface program.
- r class**
Remove the printer from an existing class. Removing the last remaining member of a class causes the class to be deleted. A destination cannot be removed if there are pending requests to that destination. In that case, you should use `lpmove` or `cancel` to move or delete the pending requests.
- v device**
Change the device for the printer. If this is the only modification, then this may be done even while `lpd` is running.

These examples are based on the `lp` configurations created in previous examples:

- Add printer `pr2` to class `c11`:

```
/usr/lib/lpadmin -ppr2 -cc11
```

- Change the interface program of the `pr2` to the model `prx` interface, change its device to `/dev/tty0`, and add it to a new class called `c12`:

```
/usr/lib/lpadmin -ppr2 -mprx -v/dev/tty0 -cc12
```

Note that printers `pr2` and `pr3` now use different interface programs even though `pr3` was originally created with the same interface as `pr2`. Printer `pr2` is now a member of two classes.

- Add printer `pr1` to class `c12`:

```
/usr/lib/lpadmin -ppr1 -cc12
```

The members of class `c12` are now `pr2` and `pr1`, in that order. Requests routed to class `c12` will be serviced by `pr2` if both `pr2` and `pr1` are ready to print; otherwise, they will be printed by whichever one is next ready to print.

- Remove printers `pr2` and `pr3` from class `c11`:

```
/usr/lib/lpadmin -ppr2 -rc11  
/usr/lib/lpadmin -ppr3 -rc11
```

Because `pr3` was the last remaining member of class `c11`, the class is removed.

- Add `pr3` to a new class called `c13`:

```
/usr/lib/lpadmin -ppr3 -cc13
```

5.5.3 Altering the system default destination

You can change or specify the system default destination even when `lpsched` is running. You can do this using the `lpadmin` command with the `-d` flag option. The form of the command is

```
lpadmin -d[dest]
```

The destination *dest* may be omitted; if so, then no destination is established as the system default.

Here are some examples of how default destinations may be specified:

- Establish class `c11` as the system default destination:

`/usr/lib/lpadmin -dc11`

- Establish no default destination:

`/usr/lib/lpadmin -d`

5.5.4 Removing destinations

You can use `lpadmin` to remove classes and printers only if there are no pending requests routed to them. You must either use `cancel` to cancel pending requests or use `lpmove` to move pending requests to other destinations before you can remove destinations. If the removed destination is the system default destination, the system will have no default destination until you specify a new default destination. When the last remaining member of a class is removed, the class is also removed. In contrast, removing a class never implies removing printers (see the third example, following).

The form of the `lpadmin` command used to remove destinations is

`lpadmin -xdest`

The destination being removed must be specified, and no other options are allowed.

Here are some examples of how printer destinations may be removed:

- Make printer `pr1` the system default destination:

`/usr/lib/lpadmin -dpr1`

Then remove printer `pr1`:

`/usr/lib/lpadmin -xpr1`

Now there is no system default destination.

- Remove printer `pr2`:

`/usr/lib/lpadmin -xpr2`

Class `c12` is also removed because `pr2` was its only member.

- Remove class `c13`:

`/usr/lib/lpadmin -xc13`

Class `c13` is removed, but printer `pr3` remains.

5.6 Setting up an Imagewriter or Laserwriter printer

The steps you take to add an Imagewriter or Laserwriter printer to your system are given here. In addition, an example of setting up a printer in general is provided so that you can set up other printers.

5.6.1 Adding an ImageWriter II printer

A/UX is already set up to use an Apple ImageWriter or an ImageWriter II as a line printer. All you have to do is connect the printer to the computer printer port and use the `lp` utility to spool and print your files. This short section describes how.

If you have any questions about unpacking or setting up your printer, refer to the owner's guide that came with it.

Printer name: The default printer name is `iw2`. See `lpadmin (1M)` in *A/UX System Administrator's Reference* for information about using or changing the printer name.

1. Connect the computer to your printer with the appropriate cable.

For the ImageWriter II, this requires a Macintosh Plus-to-ImageWriter II cable (use part 590-0552, 590-0340, or M0187). Plug one end of the printer cable to the serial interface socket at the back of the ImageWriter II, and plug the other end into the printer port identified by the printer icon on the back of the Macintosh II.

Adding an ImageWriter: If you're attaching the older ImageWriter model printer, you won't be able to use a Macintosh Plus-to-ImageWriter II cable. Instead, make a printer cable by connecting a Macintosh Plus adapter cable (part M0189 or 590-0341) to an ImageWriter cable (part M0150 or 590-0169). That is, attach the DB-9 connector of the ImageWriter cable to the DB-9 connector of the Macintosh Plus adapter cable. Then plug the DB-25 connector of the ImageWriter cable into the serial interface socket on the ImageWriter. Finally, plug the mini-8 connector of the Macintosh Plus adapter cable into the printer port on the Macintosh II. The rest of the instructions in this section apply to the earlier ImageWriter model as well as to the ImageWriter II.

2. Turn the printer on and make sure its green Select light is on.

Paper must be loaded into the printer before the Select light turns on. Because A/UX prints a cover page with each job, you should use either the forms tractor with fanfold paper or a cut-sheet feeder for separate sheets of paper. Otherwise, you'll need to feed at least two pages through the printer by hand for each print job spooled by the `lp` utility.

3. Bring A/UX up to multi-user mode if it isn't already.

If the `login:` prompt appears on your console, then your system is already in multi-user mode. If your system is in single-user mode, or if the root user's command prompt appears on your console and you're not sure if the system is in multi-user mode, type the command

```
init 2
```

and press RETURN. This ensures that A/UX is in multi-user mode. If you're prompted to check the file systems, enter `n`.

4. Log in as the root user.

Provide your password when prompted and press RETURN to accept the `mac2` terminal type.

5. Start the printer scheduler running by entering

```
/usr/lib/lpsched
```

6. Enter the command `lp /etc/motd` to test the printer.

Enter

```
lp /etc/motd
```

and press RETURN. A/UX should display its command prompt, and within a minute the ImageWriter II should print a cover page and then the file `/etc/motd`, the message-of-the-day login banner that users normally see when they log in. If this doesn't happen, check the printer and the earlier section "Stopping and Starting the Scheduler" to verify that the scheduler is running properly.

7. Edit the file `/etc/inittab` so that the scheduler is started automatically each time the system enters multi-user mode.

Change

```
lp:2:off:/usr/lib/lpsched >/dev/syscon 2>&1
```

to

```
lp:2:once:/usr/lib/lpsched >/dev/syscon 2>&1
```

5.6.2 Adding a LaserWriter printer

Apple LaserWriter printers can be used as intelligent printers for producing near-typeset quality text and graphics. (See *A/UX Text Processing Tools* for details.)

If you have any questions about unpacking or setting up your particular LaserWriter, refer to the owner's guide that came with your printer.

1. Put together a printer cable.

Currently, A/UX supports LaserWriters as serial devices. If you have a LaserWriter I, you will connect the printer to the Macintosh II using either the printer's 9-pin socket or its 25-pin socket. If you have a LaserWriter II, you must use its only socket, the 25-pin socket, so follow the instructions for putting together a cable for that socket.

Put together either of these cables:

- ☐ To connect to the 25-pin socket of a LaserWriter I or II, make a cable by connecting a Macintosh Plus adapter cable (part 590-0341 or M0189) to an ImageWriter cable (part 590-0169 or M0150); that is, attach the DB-9 connector of the ImageWriter cable to the DB-9 connector on the Macintosh Plus adapter cable.
- ☐ To connect to the 9-pin socket of a LaserWriter, make a cable by connecting a Macintosh Plus adapter cable (part 590-0341 or M0189) to a Modem cable (part 590-0197 or M0170); that is, attach the DB-9 connector of the Modem cable to the DB-9 connector of the Macintosh Plus adapter cable.

If you have a LaserWriter II NT or NTX, connect your cable's mini-8 connector to the printer port on the Macintosh II.

Plug the cable's circular connector into the Macintosh II printer port.

2. If you have a LaserWriter II NTX, you must also attach a SCSI cable between the printer's SCSI port and a SCSI port on the Macintosh II. See your printer owner's manual if you need further instructions on this.
3. With your LaserWriter *switched off*, connect the free end of your cable to the 25-pin or 9-pin socket of the printer.

Turn off the power to your LaserWriter. If you made your cable with a DB-25 connector, plug that connector into the printer's 25-pin socket. If you made a cable with a DB-9 connector, plug the connector into the printer's 9-pin socket.

4. Change the printer's mode switch to 9600 baud.

Refer to your printer's owner's guide for instructions on changing the LaserWriter from its preset AppleTalk communication mode to its 9600-baud mode.

5. Turn the printer on and make sure the green Ready light is on.

When you turn it on, the LaserWriter prints a test page if it's running correctly. Your printer's owner's guide contains more information about this self-test.

6. Bring A/UX up to multi-user mode if it isn't already.

If the login: prompt appears on your console, then your system is already in multi-user mode. If your system is in single-user mode, or if the root user's command prompt appears on your console and you're not sure if the system is in multi-user mode, type the command

```
init 2
```

and press RETURN. This ensures that A/UX is in multi-user mode. If you're prompted to check the file systems, enter n.

7. Log in to A/UX as the root user.

If the root user's command prompt appears on the console, you're already logged in as the root user.

However, if the `login:` prompt appears instead, type `root` and press RETURN. Provide your password when prompted and press RETURN to accept the `mac2` terminal type. The A/UX printer spooler is initially configured to print to an ImageWriter II. You need root privileges to change A/UX to print to a LaserWriter.

8. Start the printer scheduler running by entering

```
/usr/lib/lpsched
```

9. Edit the file `/etc/inittab` so that the scheduler is started automatically each time the system enters multi-user mode.

Change

```
lp:2:off:/usr/lib/lpsched >/dev/syscon 2>&1
```

to

```
lp:2:once:/usr/lib/lpsched >/dev/syscon 2>&1
```

10. Enter the following commands to configure A/UX to your LaserWriter.

The following command sequence reflects the assumption that you've kept the default printer name of `iw2` for your ImageWriter II. If you've used a different name, change `iw2` in the command line (`RM_PR iw2`) to your own printer name. (If you are unsure of your printer name, enter the command `lpstat -t` to determine it. See *A/UX Command Reference* for detailed information on `lpstat(1)`.)

The command line `ADD_LW laser tty2` gives your LaserWriter the printer name `laser`. You may enter a different name if you wish. See `lpadmin(1M)` in *A/UX System Administrator's Reference* for further information about using or changing the printer name.

Type

```
cd /usr/spool/lp
```

and press RETURN. When the root user's command prompt reappears, enter

```
RM_PR iw2
```

A/UX returns several status messages regarding the progress of this command. When the root user's command prompt appears again, enter

```
ADD_LW laser tty1
```

A/UX returns messages about the progress of this command, too.

11. Enter the command `lp /etc/motd` to test the printer.

Type

```
lp /etc/motd
```

and press RETURN. A/UX should display its command prompt, and within a minute the LaserWriter should print a cover page, then the file `/etc/motd`, the message-of-the-day login banner that users normally see when they log in.

5.6.3 If you are connecting a LaserWriter II NT or NTX

The LaserWriter II NT and NTX produce pages face-down in the paper tray, in contrast to the LaserWriter which places them face-up. This difference requires the two printers to print pages and files in different orders: the LaserWriter II NT and NTX need to print both the pages within a file, and a series of print requests they receive, in ascending order, and the LaserWriter needs to print in descending order. The printing system is set up to handle the LaserWriter. If you have a LaserWriter II NT or NTX, you need to change a variable in the printer interface file.

Here are the steps to adjust the software for a LaserWriter II NT and NTX.

1. Use a text editor to open the file
`/usr/spool/lp/model/psinterface.`

2. Find the variable `REVERSE`.
3. Set the variable `REVERSE` to the value 0 by changing the line to
`REVERSE=0`

Now the printer will print pages and files in ascending order.

5.6.4 Changing from a LaserWriter to an ImageWriter

If you've already followed the steps in the previous section, "Adding a LaserWriter Printer," and you want to change back to using an ImageWriter or ImageWriter II, follow these instructions:

1. Disconnect the printer cable from the LaserWriter and the Macintosh II.
2. Connect the Macintosh II to your ImageWriter II printer with the appropriate cable.

For an ImageWriter II you need a Macintosh Plus-to-ImageWriter II cable (part 590-0552 or M0187). Connect one end of the printer cable to the serial interface socket at the back of the ImageWriter II and the other end to the printer port on the Macintosh II.

Adding an ImageWriter: If you're attaching the older ImageWriter model printer, you won't be able to use a Macintosh Plus-to-ImageWriter II cable. Instead, make a printer cable by connecting a Macintosh Plus adapter cable (part M0189 or 590-0341) to an ImageWriter cable (part M0150 or 590-0169). That is, attach the DB-9 connector of the ImageWriter cable to the DB-9 connector of the Macintosh Plus adapter cable. (You may already have made this cable for attaching your LaserWriter.) Plug the DB-25 connector of the ImageWriter cable into the serial interface socket on the ImageWriter. Finally, plug the mini-8 connector of the Macintosh Plus adapter cable into the printer port on the Macintosh II. The rest of the instructions in this section apply to the earlier ImageWriter model as well as to the ImageWriter II.

3. Turn the printer on and make sure its green Select light is on.

Paper must be loaded into the printer before the Select light turns on. Because A/UX prints a cover page with each job, you should

use either the forms tractor with fanfold paper or a cut-sheet feeder for separate sheets of paper. Otherwise, you'll need to feed at least two pages through the printer by hand for each print job spooled by the `lp` utility.

4. Bring A/UX up to multi-user mode if it isn't already.

If the `login:` prompt appears on your console, then your system is already in multi-user mode. If your system is in single-user mode, or if the root user's command prompt appears on your console and you're not sure if the system is in multi-user mode, type the command

```
init 2
```

and press RETURN. This ensures that A/UX is in multi-user mode. If you're prompted to check the file systems, enter `n`.

5. Log in to A/UX as the root user.

If the the root user's command prompt appears on the console, you're already logged in as the root user.

However, if the `login:` prompt appears instead, type `root` and press RETURN. Type your password when prompted and press RETURN to accept the `mac2` terminal type. You'll need root privileges to use several of the following commands.

6. Enter the following commands to reconfigure A/UX to print to your ImageWriter II.

The following command sequence reflects the assumption that the printer name of your LaserWriter is `laser`. If you've used a different name, change `laser` in the command line `RM_PR laser` to your own printer name. (If you are unsure of your printer name, enter the command `lpstat -t` to determine it. See *A/UX Command Reference* for detailed information on `lpstat(1)`.)

The command line `ADD_IW iw2 printer` gives your ImageWriter II the printer name of `iw2`. You may enter a different name if you wish. See `lpadmin(1M)` in *A/UX System Administrator's Reference* for further information about using or changing the printer name.

Enter

```
cd /usr/spool/lp
```

When the root user's command prompt reappears, enter

```
RM_PR laser
```

A/UX returns several status messages regarding the progress of this command. When the root user's command prompt appears again, enter

```
ADD_IW iw2 printer
```

A/UX returns messages about the progress of this command, too.

7. Enter the command `lp /etc/motd` to test the printer.

When the root user's command prompt reappears, type

```
lp /etc/motd
```

and press RETURN. A/UX should display its command prompt, and within a minute the ImageWriter II should print a cover page, then the file `/etc/motd`, the message-of-the-day login banner that users normally see when they log in.

5.6.5 Setting up other printers

As an example of how to set up a hard-wired device for use as an `lp` printer, consider using `tty0` as printer `iw2`, assuming that the printer is an ImageWriter II. As the root user, do the following:

1. Avoid unwanted output from non-`lp` processes and ensure that `lp` can write to the device, as follows:

```
chown lp /dev/tty0
chgrp lp /dev/tty0
chmod 600 /dev/tty0
```

2. Change the file `/etc/inittab` so that `tty0` is not a login terminal. In other words, ensure that `/etc/getty` is not trying to log users in at this terminal. Change the entry for `tty0`, as follows:

```
00:2:off:/etc/getty tty0 at_9600
```

Enter the command

`init Q`

to tell the `init(1M)` process to reread the `/etc/inittab` file and remove the `getty(1M)` process from the port.

3. Check the status of `lp sched`. Enter the command

`lpstat -r`

If you see the message

`scheduler is running`

enter the command

`/usr/lib/lpshut`

4. Now introduce the printer `iw2` to `lp`. The notation `iw2` is the name of the interface program that configures the ImageWriter II. The interface programs for other printers you can use are listed in `/usr/spool/lp/model`. To introduce printer `iw2` to `lp`, enter

`/usr/lib/lpadmin -piw2 -v/dev/tty0 -miw`

5. If you are going to use at least two printers, you might consider creating at least a class of printers. To do this at the same time as you introduce `iw2`, replace the line in step 4 with

`/usr/lib/lpadmin -piw2 -v/dev/tty0 -cclass1 -miw`

Otherwise, if `iw2` has already been introduced, enter

`/usr/lib/lpadmin -piw2 -cclass1`

6. You can now let `lp sched` run. Enter

`/usr/lib/lpsched`

7. When `iw2` is created, it is initially disabled, and `lp` rejects requests routed to it. Allow `lp` to accept requests for `iw2` by entering the following:

`/usr/lib/accept iw2`

This allows the queuing of printing requests to `iw2` when `iw2` is enabled.

8. Before printing, be sure that the printer is ready to receive output, that the top of the form has been adjusted, and that the printer is online. To enable printing on iw2, give the command

```
enable iw2
```

When requests have been routed to iw2, they will begin printing.

9. You can now print. As a test, print the message-of-the-day (motd) file.

```
lp /etc/motd
```

In a few seconds, you should get a message with the destination and the request number from lp. A few seconds later, the file should start printing.

5.7 Writing printer interface programs

Every lp printer must have an interface program that does the actual printing on the device that is currently associated with the printer. Interface programs may be Bourne shell procedures, C programs, or any other executable program.

This section is intended to provide very general guidelines for writing printer interface programs. It assumes that you have a working knowledge of Bourne shell or C language programming principles. As with all programming, the best way to learn it is to study examples of code that works. The lp model interfaces supplied with A/UX are all written as Bourne shell procedures and can be found in the `/usr/spool/lp/model` directory. This directory contains interface programs for various printer models.

The home directory of lpsched is `/usr/spool/lp`. When lpsched is ready to print a user request on the printer *printer*, it invokes the appropriate interface program with a command line of the form

```
interface/printer id login-name title copies options file(s)
```

where the fields are interpreted as follows:

```
printer    name of the interface program in
            /usr/spool/lp/interface directory
```

id request ID returned by `lp`

login-name login name of the user who made the request

title optional title specified by the user

copies number of copies requested by the user

options blank-separated list of class- or printer-dependent options specified by the user

file(s) full pathname(s) of file(s) to be printed

To print a request, the `lp` scheduler, `lp sched`, invokes a printer interface program. The command line argument to the program is based on the spooler configuration and information contained in the `lp` command line entered by the user. The following example assumes a user named Smith and a system default printer called `iw2`.

The command

```
lp /etc/passwd -t"passwords" -n5 -oa -ob
    /etc/passwd /etc/group
```

causes `lp sched` to invoke the `iw2` interface program with the line

```
interface/iw2 iw2-52 smith "passwords" 5 "a b"
    /etc/passwd /etc/group
```

Each of these commands should appear on one long line. They are broken here because the page does not accommodate the long measure. You can now write a simple routine to have your interface program interpret the arguments and process the print request appropriately.

When the interface program is invoked, its standard input is `/dev/null`, and both the standard output and standard error output are directed to the printer's device. Devices are opened for reading as well as writing when file modes permit. When a device is a regular file, all output is appended to the end of the file.

Given the command line arguments and the output directed to a device, interface programs may format their output in any way you choose. Interface programs must ensure that the proper `stty` modes (terminal characteristics such as baud rate and output options) are in effect on the

output device. This may be done in a shell interface only if the device is opened for reading, as follows:

```
stty mode... <&l
```

In other words, take the standard input for the `stty` command from the device.

When printing is complete, the interface program should exit with a code indicating the success or failure of the print job. The `lpsched` scheduler interprets the exit codes generated by the Bourne shell `exit (n)` and the C statement `exit (n)` as follows:

0 The print job has been completed successfully.

1 to 127

A problem was encountered in printing this request (for example, too many nonprintable characters). The problem will not affect future print jobs. The `lpsched` scheduler mails the user a letter stating that there was an error in printing the request. The letter contains the error exit code.

greater than 127

These codes are reserved for internal use by `lpsched`. Interface programs must not exit with codes in this range. As before, `lpsched` uses mail to notify the user of the abnormal exit and exit code.

When problems occur that are likely to affect future print jobs (for example, a device filter program is missing), your interface program should be written to disable printers (with the `disable` command) so that print requests are not lost.

5.8 Using the `lp` system

Once `lp` destinations have been created, users may route output to a destination by using the `lp` command. The basic form of the `lp` command is

```
lp [options] files
```

Various options are available with the `lp` command; see `lp(1)` in *A/UX Command Reference*. You can use the request ID returned by `lp` to see if the request has been printed or to cancel the request.

The `lp` program determines the destination of a request by checking the following list in order:

- If the user specifies `-ddest` on the command line, the request is routed to *dest*.
- If the environment variable `LPDEST` is set, the request is routed to the value of `LPDEST`.
- If there is a system default destination, the request is routed there.
- Otherwise, the request is rejected.

Here are some examples of using the `lp` command.

- There are at least four ways to print the a file on the system default destination:

```
lp filename
lp < filename
cat filename | lp
lp -c filename
```

In the first method, the file is printed directly; in the last three, the file is printed indirectly. If you use the first command, the file is modified between the time the request is made and the time it is actually printed; the changes will be reflected in the output.

- Print two copies of file `abc` on printer `iw2` and title the output `myfile`:

```
pr abc | lp -diw2 -n2 -t"myfile"
```

- Print file `abc` on a Diablo 1640 printer called `jerry` in 12 pitch, and write the file to the user's terminal when printing is completed:

```
lp -djerry -o12 -w abc
```

In this example, 12 is a command to the model Diablo 1640 interface program to print output in 12-pitch mode. Other models may require different options. See `lpadmin(1M)` in *A/UX System Administrator's Reference*.

Note: The `lp` command runs with an effective user ID (EUID) of `lp`. In other words, A/UX behaves as if a user named `lp` is reading the files to be printed. Therefore, any files to be printed with a command of the form

```
lp [options] files
```

must have read permission set for *others*. If this poses a security problem, you can use the `lp` command in a pipe (`|`) or with the shell input redirect character (`<`). These two methods work because the user is feeding input to the `lp` command via the standard input. Here are two examples:

```
lp < /etc/passwd
pr abc | lp
```

5.8.1 Allowing and refusing requests

When a new destination is created, `lp` at first rejects requests that are routed to it. When you are sure that the new destination is set up correctly, you should use the `accept` command to allow `lp` to accept requests for that destination. See `accept(1M)` in *A/UX System Administrator's Reference*.

Sometimes it is necessary to prevent `lp` from routing requests to destinations. If printers have been removed or are waiting to be repaired, or if too many requests are in queue for printers, you may want to have `lp` reject requests for those destinations. The `reject` command performs this function; see `reject(1M)` in *A/UX System Administrator's Reference*. After the condition has been remedied, you should use the `accept` command to allow requests to be taken again.

The acceptance status of destinations is reported by the `-a` option of `lpstat`.

Here are some examples of how to reject and accept requests:

- Have `lp` reject requests for destination `iw2`:

```
/usr/lib/reject -r"printer iw2 needs repair" iw2
```

Any users who try to route requests to `iw2` will see the following message

```
lp -iw2 file
```

```
lp: cannot accept requests for destination "iw2"  
-- printer iw2 needs repair
```

- Allow `lp` to accept requests routed to destination `iw2`:

```
/usr/lib/accept iw2
```

5.8.2 Allowing and Inhibiting printing

The `enable` command allows the `lp` scheduler to print requests on printers. The scheduler routes requests only to the interface programs of enabled printers. By issuing the appropriate `enable` and `reject` commands, you can enable a printer and at the same time prevent further requests from being routed to it. This can be useful for testing purposes.

The `disable` command reverses the effects of the `enable` command. It prevents the scheduler from routing requests to printers, regardless of whether `lp` is allowing them to accept requests. Printers may be disabled for several reasons, including malfunctioning hardware, paper jams, and end-of-day shutdowns. If a printer is printing a request at the time it is disabled, the request will be reprinted in its entirety either on another printer (if the request was originally routed to a class of printers) or on the same one when the printer is enabled once again.

The `-c` option cancels the currently printing requests on busy printers in addition to disabling the printers. This is useful if strange output is causing a printer to behave abnormally.

Here are some examples of how to enable and disable a printer:

- Disable printer iw2 because of a paper jam:

```
disable -r"paper jam" iw2
printer "iw2" now disabled
```

- Find the status of printer iw2:

```
lpstat -piw2
printer "iw2" disabled since Jan 5 10:15 -
paper jam
```

- Re-enable iw2:

```
enable iw2
printer "iw2" now enabled
```

5.8.3 Moving requests between destinations

Occasionally, lp administrators find it useful to move output requests between destinations. For instance, when a printer is down for repairs, you may want to move all of its pending requests to a working printer. This is one use of the `lpmove` command. The other use of this command is moving specific requests to a different destination. The `lpmove` command will not move requests while the `lp` scheduler is running.

Here are some examples of how to move requests between destinations:

- Move all requests for printer abc to printer bobby:

```
/usr/lib/lpshut
/usr/lib/lpmove abc bobby
/usr/lib/lpsched
```

The names of all the moved requests are changed from abc-nnn to bobby-nnn. As a side effect, destination abc will not accept further requests.

- Move requests jerry-543 and abc-1200 to printer bobby:

```
/usr/lib/lpshut
/usr/lib/lpmove jerry-543 abc-1200 bobby
/usr/lib/lpsched
```

The two requests are now renamed bobby-543 and bobby-1200 and will be printed on bobby.

5.8.4 Cancelling requests

To cancel lp requests, use the `cancel` command. The `cancel` command can take two types of arguments: request IDs and printer names. Requests identified by request IDs are canceled. If you use a printer name as the argument to `cancel`, all jobs currently printing on the named printers are canceled. Both types of arguments may be intermixed. See `lp(1)` in *A/UX Command Reference*.

Here is an example of how to cancel printing:

- Cancel the request that is now printing on printer bobby:

```
cancel bobby
```

If the user who cancels a request is not the user who made it, mail is sent to the owner of the request. The lp scheduler allows any user to cancel requests, thus eliminating the need for users to find lp administrators when unusual output should be stopped.

5.9 lp system troubleshooting

The lp system problems encountered most frequently are explained here, along with their solutions.

5.9.1 Problems starting lpsched

The `lpsched` scheduler is usually invoked by the `init(1M)` process when A/UX enters multi-user mode. The invocation is a two-step process:

- The `/etc/rc` script runs `rm` to remove the `SCHEDLOCK` file in the `/usr/spool/lp` directory.
- The `init` process invokes `lpsched`.

The purpose of the `SCHEDLOCK` file is to prevent more than one invocation of `lpsched` from running simultaneously. If two (or more) copies of `lpsched` are running at the same time, there is contention over system resources, resulting in confused spooler behavior and failure to print files.

When `lpsched` finds something wrong in the lp system, it attempts to mail an error message to the root user and make an entry in the

/usr/spool/lp/log file. The SCHEDLOCK file is not removed under these conditions, because invoking lpsched again without clearing the trouble is likely to produce the same error conditions.

5.9.2 Restarting lpsched

1. When lpsched stops due to error conditions:

- ☐ Check the root user's mail for correspondence from lp.
- ☐ Check /usr/spool/lp/log for error messages.
- ☐ Use `lpstat -t` to check the spooler status for additional messages about individual printers.
- ☐ Use the `ps -ulp` command to determine if multiple copies of lpsched are running. (The status command `lpstat` will not report multiple copies of lpsched.) Write down the process ID of each lpsched you find.

2. Clear the error conditions:

- ☐ If lpsched's messages indicate damaged spooler configuration files (see "lp System Files"), use the `lpadmin` command to remake the lp system (see "Configuring the lp System," earlier in this chapter).
- ☐ Use the `kill(1)` command to kill *all* of the lpsched processes. (See `kill(1)` in *A/UX Command Reference*.)
- ☐ Clear any other error conditions indicated by the error messages.

3. Restart lpsched with the commands

```
rm /usr/spool/lp/SCHEDLOCK
/usr/lib/lpsched
```

Use the `lpstat -t` command to check the status of the entire lp system.

4. If everything appears normal in the `lpstat` report, perform the ultimate test: print a file.

5.9.3 Repairing a damaged outputq file

The lp system keeps all queue data in the binary file `/usr/spool/lp/outputq`. If this file is damaged, the spooler does not run correctly, and old job files remain in the subdirectories of `/usr/spool/lp/request`. To correct this condition:

1. Use the `fsck` utility to check the file system. (See Chapter 8, "Checking the A/UX File System: `fsck`.")
2. Use the `/usr/lib/lpshut` command to stop the lp spooler.
3. Remove the contents of the directories under `/usr/spool/lp/request`, but *do not* remove the directories themselves. Use the `rm ./ *` command, but with caution! Use `pwd` to be sure you are in the directory you think you are in!
4. Nullify the corrupted `outputq` file with the command

```
cp /dev/null /usr/spool/lp/outputq
```
5. Use the `/usr/lib/lpsched` command to restart the spooler.

5.9.4 lp system files

This section describes the system files used by lp.

`/usr/spool/lp/class`

A directory containing one text file for each printer class. The filename corresponds to the class. Each class file contains the names of the printers belonging to the class.

`/usr/spool/lp/default`

A text file containing the name of the system default printer; empty if there is no default printer or destination.

`/usr/spool/lp/log`

A text file containing a record of all printing requests.

`/usr/spool/lp/FIFO`

A named pipe, readable and writable only by lp. Any lp command may write to this file, but only `lpsched` may read it.

/usr/spool/lp/interface/*printer*

The *printer* field is the name of a particular printer interface program in the /usr/spool/lp/interface directory. All files in this directory should be executable by lp only.

/usr/spool/lp/log, /usr/spool/lp/oldlog

The file /usr/spool/lp/log is a record of printing requests made during each run of lpsched. Each time lpsched is started, it copies /usr/spool/lp/log to /usr/spool/lp/oldlog. Then it truncates /usr/spool/lp/log.

/usr/spool/lp/member

A directory containing one text file for each printer. The file name corresponds to the printer name. The file contains the name of the device file in the /dev directory that corresponds to the printer.

/usr/spool/lp/model

A directory containing sample printer interface programs (Bourne shell scripts).

/usr/spool/lp/outputq

A binary data file that holds the lp request queue information.

/usr/spool/lp/pstatus

A binary data file that contains status information (whether a printer is enabled or disabled) for each printer.

/usr/spool/lp/qstatus

A binary data file that contains the acceptance status (whether a printer is accepting or rejecting requests) for each printer.

/usr/spool/lp/request

A directory containing subdirectories named for each destination (class or printer) known to the lp system. The subdirectories are used for temporary storage of spooler commands and print requests (text).

/usr/spool/lp/SCHEDLOCK

A file designed to prevent more than one invocation of lpsched from running simultaneously. See "Stopping and Starting the Scheduler," earlier in this chapter.

`/usr/spool/lp/seqfile`

A text file containing the sequence number assigned to the last request printed. The number always is in the range 1-9999.

`/usr/spool/lp/OUTQLOCK`

`/usr/spool/lp/PSTATLOCK`

`/usr/spool/lp/QSTATLOCK`

`/usr/spool/lp/SEQLOCK`

Various lock files for preventing lp system commands from modifying data in the data files described above. Each file has an expiration time, after which any lp system command may remove the lock file and then modify the previously locked data file. These lock files and SCHEDLOCK operate according to similar principles.

5.9.5 lp system command permissions

All lp system utilities with the exception of lpsched should be owned by lp with the set user ID (SUID) bit turned on (see `chmod(1)` and `chown(1)` in *A/UX Command Reference*). The lpsched scheduler may be owned either by the root user or by lp.

PLACE TAB
HERE

CHECKING THE
FILE SYSTEM

BACK OF TAB

Chapter 8

Checking the A/UX File System: **fsck**

Contents

1. Introduction to fsck	1
2. Overview of the A/UX file system	2
2.1 Partitions, file systems, and hierarchies	2
2.2 Bytes, blocks, and disk allocation	3
2.3 Inodes	3
2.4 Direct and indirect blocks	5
2.5 More on inodes	7
2.6 Starting from the top	9
2.7 Inode location	10
2.8 The superblock and the i-list	11
2.8.1 The superblock and the free list	11
2.8.2 The i-list	13
2.9 Block I/O	14
2.10 The buffer cache	14
2.11 Special files and the /dev directory	15
2.11.1 The contents of device inodes	16
3. How fsck works	17
3.1 File system updates	18
3.2 fsck phases	21
3.2.1 Phase 1: check blocks and sizes	21
3.2.2 Phase 2: check pathnames	21
3.2.3 Phase 3: check connectivity	21
3.2.4 Phase 4: check reference counts	21
3.2.5 Phase 5: check free list	21
3.2.6 Phase 6: salvage free list	22
4. Using fsck	22
4.1 When to use fsck	22
4.2 fsck options	23

4.3	fsck: a sample interaction	25
4.4	Multiple file systems and fsck	26
5.	fsck messages	29
5.1	fsck initialization phase messages	29
5.1.1	fsck option errors	29
5.1.2	Memory request errors	30
5.1.3	Errors in opening files	30
5.1.4	File status errors	30
5.1.5	File system size and inode list size	31
5.1.6	Scratch file errors	31
5.1.7	Interactive messages	31
5.2	Phase 1: check blocks and sizes	33
5.2.1	Inode type errors	33
5.2.2	Zero-link-count table errors	34
5.2.3	Bad or duplicate blocks	34
5.2.4	Inode size errors	37
5.2.5	Inode format errors	38
5.3	Phase 1B: rescan for more duplicates	39
5.4	Phase 2: check pathnames	39
5.4.1	Root inode mode and status errors	39
5.4.2	Directory inode pointers range errors	40
5.4.3	Directory entries pointing to bad inodes	40
5.5	Phase 3: check connectivity	42
5.5.1	Unreferenced directories	42
5.6	Phase 4: check reference counts	43
5.6.1	Unreferenced files	43
5.6.2	lost+found directory errors	43
5.6.3	Incorrect free inode counts	44
5.6.4	Unreferenced files/directories	45
5.6.5	Bad and duplicate blocks in files and directories	46
5.7	Phase 5: check free list	47
5.8	Phase 6: salvage free list	49
5.9	Cleanup	50

Figures

Figure 8-1. I-number relationships	4
Figure 8-2. Indirect blocks	6
Figure 8-3. Additional information in an inode	8
Figure 8-4. File-directory connection through inodes	9
Figure 8-5. I-numbers and inode locations	10
Figure 8-6. The superblock and the free list	13
Figure 8-7. How <code>fsck</code> decides whether to check a file system	27
Figure 8-8. A description of sample entries in <code>/etc/fstab</code>	28

BLANK PAGE

Chapter 8

Checking the A/UX File System: `fsck`

1. Introduction to `fsck`

The `fsck` program (for "file system check") locates and resolves inconsistencies within a file system. It is intended for use during normal booting procedures and at any other time that the system administrator suspects inconsistencies within the file system (such as immediately following a system crash).

The A/UX operating system treats almost everything in its environment as a file. To operate on a file in the A/UX environment, you need only refer to it by name. The general function of the A/UX file system is to

- support the seemingly simple interface on A/UX mass-storage media (disks, tapes, and tape cartridges)
- permit the kernel to find data on the disk
- load the data into main memory
- periodically update the disk with the modifications performed on the data in main memory

Sometimes this updating fails, usually because of power failures or improper system shutdown, and inconsistencies within the file system result. Fortunately, in most cases you can resolve these inconsistencies by using the A/UX `fsck` program.

The `fsck` program checks the location of files on disk and uses redundancies and known parameters to resolve inconsistencies. A **known parameter** is information about the file system that does not change, such as the number of characters per block or the number of blocks in a disk. A **redundancy** is information that the system maintains in more than one place, such as the size of each file and the number of blocks not currently in use.

In many cases, `fsck` can fix the damage. Sometimes, however, `fsck` can only report cryptic messages about the damage that has been done.

In these cases, a system administrator who knows the structure and functioning of the file system must resolve the problem. The goal of this chapter is to provide you with this knowledge.

Section 2 of this chapter describes the structure of the A/UX file handling system. Section 3 discusses how `fsck` operates and when you should run it. Section 4 discusses how to use `fsck`. Section 5 describes `fsck`'s phases and error messages.

Before you begin this chapter you should know how to use the commands `ls`, `rm`, `cp`, `mv`, `cd`, and `ln` (for more information about these commands, refer to Chapter 4, "The A/UX File System," in *Getting Started with A/UX*). You should also be able to bring the system to single-user or multi-user status (see Chapter 2, "System Startup and Shutdown," for more information).

2. Overview of the A/UX file system

It is important to understand the organization of the A/UX file system and some of the commands that manipulate this organization before you begin to work with `fsck`. This section gives you a brief overview of the relevant file and directory information.

2.1 Partitions, file systems, and hierarchies

The A/UX file-handling system is the complete set of data structures, commands, and subroutines used to manipulate data stored on a physical device. On the A/UX system, a physical storage device (usually a disk) is divided into logical devices called partitions, each of which may contain an A/UX file system. You can use the `dp` command to establish or remove partitions; see `dp(1M)` and `gd(7)` in *A/UX System Administrator's Reference*, and `bzb(4)` and `dpme(4)` in *A/UX Programmer's Reference*.

A file system is a logical device that contains the data structures (directories, files, and inodes, among others) that implement all or part of the A/UX directory hierarchy. The A/UX directory hierarchy is simply the collection of all files currently available to the system. This coincides with the collection of all files on currently mounted file systems; see `mount(1M)` in *A/UX System Administrator's Reference*. From the user's point of view, the directory hierarchy resembles an inverted tree, branching out from the root directory (`/`).

The general term *hierarchy* is used for the collection of files and subdirectories of a given directory; for instance, the `/usr` hierarchy means all files and directories under the `/usr` directory. A hierarchy coincides with the A/UX directory hierarchy only when the directory under consideration is the root directory.

2.2 Bytes, blocks, and disk allocation

A file system is divided up into units called **blocks**. A block on the disk is called a **physical block** and is a contiguous sequence of 512 bytes (characters). To speed up the disk I/O operations, A/UX handles two physical blocks at once (1024 bytes); this is called a **logical block**. A logical block is two contiguous 512-byte blocks. Each character in a file is represented by a byte of information, and each byte resides in a block.

A file smaller than one logical block, also called a **data block**, resides in one location of the disk. A file larger than this resides in different portions of the disk. Ideally, the data blocks of a file should be as near to one another as possible to optimize disk I/O operations. However, in reality, as the disk is used and files are deleted, data blocks become increasingly scattered. To handle this, inodes are made to point to each block of the file in sequence.

2.3 Inodes

Inodes contain information about files, the most important of which is a list of locations on the disk where the file's contents are to be found. Note that the inode does not point to a single location on the disk, but to several discrete locations (see the next section, "Direct and Indirect Blocks"). Figure 8-1 illustrates the relationship between the directory `/users/demo`, a file in that directory named `letter`, the i-number and inode associated with `letter`, and the disk locations where the contents of `letter` are stored.



file with 1025 characters occupies two logical blocks.

2.4 Direct and Indirect blocks

The inode contains 13 disk addresses. The first 10 addresses point to the first 20 physical blocks of the file. These blocks are the **direct data blocks**. The first 1024 characters of a file reside in the block located at the first address to which the inode points. If a file has more than 1024 characters, it continues at the second address to which the inode points. If it has more than 2048 characters, it continues at the third address, and so on, until the first 10 addresses (10240 characters) have been used.

If a file has more than 10240 characters, the 11th address given in the inode is then taken into consideration. The 11th disk address points to an **indirect block**. An indirect block contains the addresses for the next 256 logical blocks that the file can use. Figure 8-2 illustrates these connections.

If the file is larger than 138 blocks (10 blocks for the first 10 addresses, 256 for the 11th), it continues at the 12th address given in the inode. The 12th address points to a **double indirect block**, which refers to up to 256 indirect blocks that the file can use. This gives a total of 33,690,624 bytes of storage: 10 from the first 10 addresses + 256 from the 11th address + $(256 * 256)$ from the 12th address * 512 (per block) = 33,690,624 bytes.



refers to up to 256 indirect blocks, and so on. Now you have 8,623,625,216 bytes:

$$(10 + 256 + (256 * 256) + (256 * 256 * 256) * 512 = 8,623,625,216$$

This is the maximum size a file can be in a file system that has 1024-byte logical blocks. A system with a larger bytes-per-block value could have a much larger theoretical limit. But because the file size is represented in an inode by a signed 32-bit quantity, a file can never get larger, in practice, than about 2 gigabytes. A file can never be this big because the physical storage device imposes a ceiling on the maximum file size.

2.5 More on Inodes

Inodes contain information about the location of the data blocks that make up a file. Figure 8-3 illustrates the other important information that inodes contain about a file.

2.8 The superblock and the i-list

The system keeps track of all free inodes and all active inodes in a twofold way. First, it uses the second block in the file system (address 1) to store information about the file system itself. This block is called the **superblock**. (Address 0, the first block, is used to maintain disk information.) Second, it uses a list called the **i-list**, discussed later.

2.8.1 The superblock and the free i-list

Among other things, the superblock contains information about the size (in blocks) of the i-list (explained in the next section), the size (in blocks) of the entire file system, the total number of free blocks, and, most importantly, the free inode list and the total number of free inodes.

The superblock also contains the beginning of a linked list that maintains information about the free blocks. This list is called the **free list**. The free list is a list of addresses of all the free blocks in the file system. The free list is maintained in discrete chunks, in the following way. The first 50 addresses in the free list are contained within the superblock; this is usually referred to as the **superblock free list**. Forty-nine of these addresses are simply the addresses of the 49 next free blocks. The 50th address is the address of a block containing 50 more addresses. There is no standard name for this block of 50 addresses; the name **free list link block** is used here.

Each free list link block contains 49 direct addresses and 1 indirect address, until the end of the free list is reached. (A link block could, in theory, hold as many as 128 addresses, but only 50 are stored in a free list link block so that a link block can be read entirely into the superblock when the addresses it contains are needed.) The beginnings of a free list are illustrated in Figure 8-6.

file inodes. Each inode is a 64-byte structure containing information about the physical location of the file associated with it.

The important thing is that the superblock and the i-list are located not only on disk but also in main memory, and the main memory versions are the updated ones. This is important when it comes to the interaction between main and secondary memory.

2.9 Block I/O

It would be both risky and expensive to keep all data in main memory (also called **primary memory**, **in-core memory**, or **RAM**). Instead, most files are kept in secondary memory (for example, a disk), and the system brings them into main memory as necessary. If you modify files, the system writes the modified version back into secondary memory for future use.

When a program reads data from or writes data to a tape or a disk, the system extracts blocks of 1024 bytes and brings them into main memory. It would be impractical and even unsafe to bring data into main memory without imposing limits and some degree of organization on the amount of data transferred. It would also be highly inefficient to do physical I/O operations whenever data is transferred from primary to secondary memory and vice versa. For that reason, the system maintains a list of **buffers** for each device. This buffer pool is said to constitute a **data cache** for the device in question.

2.10 The buffer cache

When a program asks the system to read data from a file, the system first searches the cache for the desired block.

If the block is found (when, for instance, the system opens a file that is already open), the data is made available to the requesting process without a physical I/O read operation. If the data is not found in the cache, the buffer that has been unused for the greatest period of time is renamed, and data is transferred into it from the disk and made available.

When a process writes a file, the operation occurs in reverse order. A write request first writes data to the buffer cache. Things are written to disk only when the cache is full. Therefore, information about bad

writes refers generally to unusual bad writes to the buffer, and bad disk writes are generally reported too late to prevent the file system from being corrupted.

2.11 Special files and the /dev directory

There are several types of files in A/UX: regular files, directories, special files (device files), sockets, symbolic links, and named pipes. In the beginning days of UNIX, only three types of files existed: regular files, directories, and devices. In this context, device files were special and were given the name "special" files. The addition of other file types makes the name no longer appropriate, yet the name holds.

When you use the `ls -l` command to list your files, the system response looks like this:

```
-rw-rw---- 1 groupname 13 Sep 25 11:28 file
drwxrwx--- 2 groupname 32 Sep 25 11:28 directory
```

For regular files, such as the first one listed in the example, the first character in the permissions field is `-`. In the case of directories, this character is always `d`. However, suppose you list `/dev/rdisk/c0d0s0` and `/dev/dsk/c0d0s0` by giving the command

```
ls -l /dev/rdisk/c0d0s0 /dev/dsk/c0d0s0
```

In response, the system displays

```
crw----- 2 bin 24, 0 May 25 1987 /dev/rdisk/c0d0s0
brw----- 2 bin 24, 0 May 25 1987 /dev/dsk/c0d0s0
```

The first character in the permissions field of `/dev/rdisk/c0d0s0` is `c`, and the first character in the permissions field of `/dev/dsk/c0d0s0` is `b`. Either a `b` or a `c` in this position indicates a special (device) file.

Now suppose you list `/usr/spool/lp/FIFO` by giving the command

```
ls -l /usr/spool/lp/FIFO
```

The `p` in the first field tells you that the file is a pipe.

```
prw----- 1 lp lp 0 Oct 21 1987 /usr/spool/lp/FIFO
```

The other two types of file have their own symbols: `l` (symbolic links) and `s` (sockets).

2.11.1 The contents of device inodes

The files in the `/dev` directory are all special files that the system uses to select a device driver for performing physical I/O.

These files are actually just names and inodes with no associated data on disk (and thus a size of zero bytes). Instead of storing information about the number of bytes in a file, these inodes contain a major and minor number for each special file. These are the numbers you see displayed after you give the `ls -l` command shown in the previous section, "Special Files and the `/dev` Directory."

A device driver is a program that controls the actual physical I/O to the devices listed in the `/dev` directory. However, the device driver itself doesn't reside in the `/dev` directory; rather, it is compiled directly into the kernel.

There is a different device driver for each kind of device (disk, tape, and so on). The system uses the major number to access the correct device driver. The minor number is passed to the driver, which uses this argument to select the correct physical device.

In summary, the system takes the following steps in response to requests to open special files (such as `fsck` may make):

- looks in `/dev` directory for a file with the requested name
- gets the i-number associated with the filename
- finds the inode specified by the i-number
- gets the major number stored in the inode
- uses this number to select the appropriate device driver
- passes the minor number to the device driver

The driver then uses the minor number to select the correct physical device (the proper partition, in the case of the disk device).

As you may have already guessed, devices (and therefore device drivers and their corresponding special files) come in two forms, `b` (block) and `c` (character)—hence the `b` and `c` in the `ls -l` listing.

These names refer to the method of I/O used with each type of device.

Block devices such as disks use the block I/O buffer cache mentioned previously and are thus written to, and read from, one block at a time. Character devices such as terminals, line printers, and modems are written to, and read from, one character at a time. Another name for character devices is *raw* devices.

Disk partitions are peculiar beasts because each partition is associated with both a character and a block device driver and thus with two special files in the `/dev` directory. For this reason, you can access disk partitions in two ways. They're normally accessed as block devices through the A/UX directory hierarchy. But certain programs that access disks, such as `dump.bsd`, `dd`, `volcopy`, and `fsck`, run faster when accessing the disk as a character device. For example,

```
fsck /dev/rdisk/c0d0s0
```

is faster than

```
fsck /dev/dsk/c0d0s0)
```

The listing

```
crw----- 1 bin  24, 0 May 25 1987 /dev/rdisk/c0d0s0
brw----- 1 bin  24, 0 May 25 1987 /dev/dsk/c0d0s0
```

is clearly a listing of a character (raw) device and a block device. The first has a `c` and the second a `b` as the first entry in the permissions field. Thus, the same device can have two different interfaces.

3. How `fsck` works

As you open, create, and modify files, the system keeps track of all pertinent information about them. This information—including block sizes, their i-numbers, active and free inodes in the file system, and total number of active, used, and free blocks—is maintained and updated in main memory. If the system crashes or becomes corrupt for any reason, the various file systems will probably become inconsistent. This inconsistency arises because the information in the main memory superblock was not written to disk before the problem, whereas the table of active inodes probably was, since the table occurred is written to disk whenever a file is released.

A system crash is not the only cause of file system corruption. Time and normal wear and tear produce inconsistencies that must be detected and corrected.

The `fsck` program works by comparing one or more items of information to one or more items of equivalent information. For instance, it compares the number of free blocks available to the number of total blocks in the file system minus the number of blocks in use. If the two numbers are not equal, `fsck` generates an error message. This kind of error is due to an inconsistent update or one that was performed out of order. To understand the problems `fsck` is designed to solve, you need to understand these updates.

3.1 File system updates

This section describes the various file system updates the system performs every time you create, modify, or remove a file. There are five types of file system updates:

Superblock

Contains information about the size of the file system and inode list, part of the free list, a count of free blocks, a count of free inodes, and part of the free inode list.

A mounted file system's superblock is written to disk whenever the file system is unmounted or a `sync(1M)` command is issued. The exception to this is that the root file system is always mounted.

The superblock of a file system is prone to inconsistency because every change to the blocks or inodes of the file system modifies the superblock.

Inode

Contains information about the inode's type (which may be directory, data, or special), the number of directory entries linked to it, the list of blocks claimed by the inode, and the inode's size. An inode is written to disk when the file associated with it is closed. In fact, all in-core blocks are also written to disk when a `sync` system call is issued; thus, the period of danger when inconsistencies can appear is reduced to that between `sync` calls.

Indirect blocks

Can be one of three types: single-indirect, double-indirect, or triple-indirect.

Indirect blocks, as well as the first 10 blocks of a file, are written to disk whenever they have been modified or released by the operating system. More precisely, they are queued in the buffer cache for eventual writing. Physical I/O is deferred until the A/UX operating system needs the buffer or a `sync` command is issued.

Inconsistencies in an indirect block directly affect the inode that owns it.

You can check the following inconsistencies: blocks already claimed by another inode and block numbers outside the range of the file system.

Data block

Written to disk whenever it has been modified.

There are two types of data blocks: plain and directory. Plain data blocks contain the information stored in a file. Directory data blocks contain directory entries.

The `fsck` program does not attempt to check the validity of the contents of a plain data block.

In contrast, `fsck` checks each directory data block for inconsistencies involving the following:

- directory inode numbers pointing to unallocated inodes
- directory inode numbers greater than the number of inodes in the file system
- incorrect directory inode numbers for the dot files (`.`) and (`..`)
- directories disconnected from the file system

If a directory entry inode number (that is, a file's inode) points to an unallocated inode, `fsck` may remove that directory entry, depending on how `fsck` was invoked. (For instance, it removes the entry if invoked with the `y` flag option.) See "`fsck` Messages." This condition might occur when the data blocks

containing the directory entries are modified and written out while the inode is not yet written out.

If a directory entry inode number points beyond the end of the inode list, `fsck` may remove that directory entry. This condition occurs if bad data is written into a directory data block.

The directory inode number entry for the dot file `.` should be the first entry in the directory data block. Its value should be equal to the inode number for the directory data block.

The directory inode number entry for the dot file `..` should be the second entry in the directory data block. Its value should be equal to the inode number for the parent of the directory entry (or the inode number of the directory data block if the directory is the root directory).

If the directory inode numbers are incorrect, `fsck` may replace them with the correct values.

The `fsck` program checks the general connectivity of the file system. If directories are found not to be linked into the file system, `fsck` links the directory back into the file system's `lost+found` directory. This condition can be caused if inodes are written to disk but the corresponding directory data blocks are not.

Free list

The system updates the free list when a file has been deleted or when a file has been enlarged past a block boundary. As described previously (see Figure 8-6), the free list begins in the superblock, which contains 49 addresses of blocks available for data storage plus the address of the next free list link block. When the first 49 addresses are used up, the kernel uses the address of the next free list link block to re-initialize the free list in the superblock. As long as disk storage is available, this new list will contain the addresses of the next 49 available free blocks plus the address of the next free list link block.

Free list link blocks are chained from the superblock. Therefore, inconsistencies in free list link blocks ultimately affect the superblock.

You can check the following inconsistencies : a list count

outside of range, block numbers outside of range, and blocks already associated with the file system.

3.2 fsck phases

There are six file system check phases in `fsck` (one of which is generally optional) as well as an initialization phase. Each phase of the `fsck` program passes through the whole file system. If you invoke `fsck` without a device name in the command line, `fsck` repeats all its phases for all devices.

3.2.1 Phase 1: check blocks and sizes

The `fsck` program checks the inode list. In this phase, `fsck` may discover error conditions that result from checking inode types, setting up the zero-link-count table, checking inode block numbers for bad or duplicate blocks, checking inode size, and checking inode format. Phase 1B runs only if any duplicate blocks (that is, blocks that belong to multiple inodes) are found.

3.2.2 Phase 2: check pathnames

The `fsck` program removes directory entries pointing to inodes that have error conditions from phase 1 and phase 1B. In this phase, `fsck` may discover error conditions that result from root inode mode and status, directory inode pointers out of range, and directory entries pointing to bad inodes.

3.2.3 Phase 3: check connectivity

The `fsck` program checks the directory connectivity seen in phase 2. In this phase, `fsck` may discover error conditions that result from unreferenced directories and missing or full `lost+found` directories.

3.2.4 Phase 4: check reference counts

The `fsck` program reports messages that result from unreferenced files; a missing or full `lost+found` directory; incorrect link counts for files, directories, or special files; unreferenced files and directories; bad and duplicate blocks in files and directories; and incorrect total free inode counts.

3.2.5 Phase 5: check free list

The `fsck` program checks each free list link block for a list count out of range, for block numbers out of range, and for blocks already allocated within the file system. A check is made to see that all the blocks in the file system were found.

3.2.6 Phase 6: salvage free list

The `fsck` program concerns itself with the reconstruction of the free list. It lists error conditions resulting from the blocks-to-skip and blocks-per-cylinder values.

4. Using `fsck`

You can use several options with the `fsck` utility, depending on whether you want to check the root file system or auxiliary file systems.

4.1 When to use `fsck`

Any file system inconsistency will be made worse if you continue to use the file system (thus modifying it further) without running `fsck`. Because it is so important to keep your file systems consistent, the `fsck` program is programmed into the system startup procedure (see Chapter 2, "System Startup and Shutdown") and is automatically run each time you start up A/UX, providing you respond yes to the prompt regarding checking the file system.

You can also invoke `fsck` at any time during a session by bringing the system down to single-user mode and entering

```
fsck [options] [file-systems]
```

You can specify *options* to direct `fsck` to run in a different way; see `fsck(1M)` in *A/UX System Administrator's Reference* for a complete list of options. You can specify *file-systems* to run `fsck` on a different file system. File system names are defined in the file `/etc/fstab`; see `fstab(4)` in *A/UX Programmer's Reference* for details. If you enter `fsck` without options or file system names, it will run on all file systems.

Note that the file system on which `fsck` is running should be unmounted, or at least no writes should occur while `fsck` is running. This is important because `fsck` performs more than one pass on the file system. If the system is modified from pass to pass, the results are unpredictable.

When `fsck` finds an inconsistency in a file system, it informs you with a message such as

```
/dev/dsk/c0d0s0 POSSIBLE FILE SIZE ERROR I=2405
```


The message can also look like this:

```
/dev/dsk/c0d0s0 FREE INODE COUNT WRONG IN SUPERBLK  
FIX?
```

The second message illustrates one of `fsck`'s interactive error messages. The program performs the corrective action only if you type `y` to confirm that it should do so. Otherwise, it will either continue or terminate, depending on the nature of the problem encountered.

4.2 `fsck` options

You can specify the following flag options on the `fsck` command line:

- y Automatic yes. The `y` flag option automatically provides a yes response to all questions that `fsck` asks.
- n Automatic no. The `n` flag option automatically provides a no response to all questions that `fsck` asks. The file system is not opened for writing.
- f Fast check. This causes `fsck` to check blocks and sizes (phase 1) and check the free list (phase 5). The free list is reconstructed (phase 6) if necessary.
- p [*pass-to-start*]
Automatic fix. If you give the `p` flag option, `fsck` automatically fixes those file system inconsistencies that it can fix without your assistance. The optional *pass-to-start* argument specifies the starting pass number. You cannot give `fsck` a starting pass number except as an argument to `p`. The starting pass number limits which file systems `fsck` checks; `fsck` looks in the `/etc/fstab` file and checks those file systems with pass numbers equal to or greater than the *pass-to-start* argument. If you don't specify a pass number, the default is 1.
- q Quiet `fsck`. If you give the `q` option, `fsck` does not print size-check messages in phase 1. Unreferenced FIFO's are silently removed. If `fsck` requires it, counts in the superblock are automatically fixed, and the free list is salvaged.
- s x
Unconditional reconstruction of the free list. The `s` option causes `fsck` to ignore the actual free list and (unconditionally)

reconstruct a new one by rewriting the super-block of the file system. The file system should be unmounted while this is done; if this is not possible, you should be careful that the system is quiescent and that it is rebooted immediately afterward. This precaution is necessary so that the old, bad in-core copy of the superblock will not continue to be used or written on the file system.

The *s x* flag option allows you to create an optimal free-list organization. The option has the following form:

-s blocks-per-cylinder:blocks-to-skip

If *x* is not given, *fsck* uses the values specified when the file system was created. If these values were not specified, then the value 400 : 7 is used.

-S x

Conditional reconstruction of the free list. This flag option is like *s x*, except that the free list is rebuilt only if no discrepancies were discovered in the file system. Using *S* forces a no response to all questions that *fsck* asks. This flag option is useful for forcing free-list reorganization on uncontaminated file systems.

-t file

Named scratch file. If *fsck* cannot obtain enough memory to keep its tables, it uses a scratch file. If the *t* option is specified, the file named in the next argument is used as the scratch file, if needed. Without the *t* flag, *fsck* prompts the operator for the name of the scratch file. The file chosen should not be on the file system being checked. If the scratch file is not a special file or did not already exist, *fsck* removes it at the end of the run.

-D [options]

Options. If the *options* argument is missing, *fsck* merely checks directories for bad blocks. The *options* argument may be any of the following:

- B** Check for and clear parity bits in filenames.
- C** Check whether all trailing characters in the filename are null.

CZ Check and write nulls in all trailing characters in the filename.

4.3 **fsck**: a sample interaction

If you bring the system down to single-user mode and enter

```
fsck -n
```

fsck starts running. Because you didn't specify a file system, **fsck** reads the file `/etc/fstab`, which contains a list of the files to be checked in this case. Also, because you invoked **fsck** with the `n` option, **fsck** assumes you're always answering no to its prompts and thus doesn't open the file system for writing. In other words, you are safe.

Note: Unless you know exactly what you're doing, always invoke **fsck** a first time with the `n` option. Read the messages and decide in advance the course of action to take before you invoke it a second time without the `n` option.

For each file system checked, you will see a screen message similar to this one:

```
/dev/dsk/c0d0s0 (NO WRITE)
File System: root Volume: 001

** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Free List
402 files 12374 blocks 950 free
```

The **fsck** program makes five passes (sometimes six) and lets you know at what phase and in what file system (in this example, `/dev/dsk/c0d0s0`) it is at any given time. Different and separate file systems are discussed in the next section, "Multiple File Systems and **fsck**."

The preceding example illustrates a routine check during which no problems or inconsistencies were found. If **fsck** finds a problem, you will see a screen message similar to this one:

```

/dev/rdisk/c0d0s1 (NO WRITE)
File System: usr Volume: 003

** phase 1 - Check blocks and sizes
POSSIBLE FILE SIZE ERROR I=1147
POSSIBLE FILE SIZE ERROR I=1195

** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Free List
1350 files 20582 blocks 18686 free

```

However, the messages can be more obscure and require some action on your part. See "fsck Messages," later in this chapter.

You can also invoke `fsck` with the `sx` or the `Sx` flag options discussed earlier. The `sx` flag option unconditionally makes a new free list, discarding the present one; the `Sx` flag option does the same only if nothing is wrong in the free list. The immediate result of these options is speeding up disk operations in busy file systems, because a new free list is likely to have more contiguous portions of free space.

The `D` flag option is useful for doing extra consistency checks on directories, and it does not prolong the process significantly. It is included in your startup `fsck` procedures. For more information about these and other options, see `fsck(1M)` in *A/UX System Administrator's Reference*.

4.4 Multiple file systems and `fsck`

File system checks occur when the system goes from single- to multi-user mode. You need to take a few steps to make sure that `fsck` automatically checks file systems other than the root file system when you respond yes to the prompt, "Do you want to check the file systems?"

Two factors determine whether a file system is checked: options given to the `fsck` command and two fields in the `/etc/fstab` file. As shipped, the system runs `fsck` during startup. The determinant fields in `/etc/fstab` file are the *type* of the file system and the *pass-number*. Figure 8-7 shows how `fsck` uses its options and these two fields to decide whether to check a file system.

5. fsck messages

When *fsck* detects an inconsistency, it reports the error condition in a message. If a response is required, *fsck* waits for it. This section explains the possible messages in each phase, the meaning of each message, the possible responses, and the related error conditions. The messages are organized by the *fsck* phase in which they can occur.

5.1 fsck Initialization phase messages

Before a file system check can be performed, certain tables have to be set up in memory, and certain files have to be opened. This is the "initialization phase" of *fsck*.

During this phase, *fsck* may discover error conditions that result from errors in command line options, memory requests, file opening, file status, file system size checks, and scratch file creation.

5.1.1 fsck option errors

The following messages may occur when you specify the *fsck* command line incorrectly. See *fsck(1M)* in *A/UX System Administrator's Reference* for further details.

c option?

c stands for any character that is not a legal flag option to *fsck*. Legal options are *y*, *n*, *s*, *S*, *q*, *D*, and *t*; *fsck* terminates on this error condition.

Bad *-t* option

The *t* option was not followed by a filename; *fsck* terminates on this error condition.

Invalid *-s* argument, defaults assumed

This is only a warning. The *s* option was not followed by 3, 4, or *blocks-per-cylinder:blocks-to-skip*. The *fsck* program assumes a default value of 400 blocks per cylinder and 9 blocks to skip.

Incompatible options: *-n* and *-s*

It's not possible to salvage the free list without modifying the file system; *fsck* terminates on this error condition.

5.1.2 Memory request errors

The following messages may occur when `fsck` detects errors in memory allocations requests. These messages may indicate a serious problem that you may not be able to solve without technical assistance.

`can't fstat standard input`

The attempt to use `fstat` as standard input failed; `fsck` terminates on this error condition.

`can't get memory`

The `fsck` program can't find the memory space it needs for its virtual memory tables; `fsck` terminates on this error condition.

5.1.3 Errors in opening files

The following messages may occur when `fsck` cannot open a file or file system.

`can't open: f`

The default file system check file `f(/etc/fstab)` cannot be opened for reading; `fsck` terminates on this error condition. Check the access modes of `f` and modify accordingly.

`can't open f`

The file system `f` cannot be opened for reading. The `fsck` program ignores this file system and continues checking the next file system given. Check the access modes of `f` and modify accordingly.

`f is not a block or character device`

The `fsck` program has been given a regular filename by mistake; `fsck` ignores this file system and continues checking the next file system given. Check the file type of `f` and modify accordingly.

5.1.4 File status errors

The following messages may occur when `fsck` cannot obtain a file's status. These errors may indicate a serious problem that you may not be able to solve without technical assistance.

`can't stat root`

The `fsck` program request for statistics about the root directory (`/`) failed; `fsck` terminates on this error condition.

can't stat *f*

The *fsck* program request for statistics about the file system *f* failed. It ignores this file system and continues checking the next file system given. Check the access modes of *f*.

5.1.5 File system size and inode list size

The file system size and inode list size are critical pieces of information to the *fsck* program. Although *fsck* can't actually check these sizes, it can check if they're within reasonable bounds. The file system size must be larger than the number of blocks used by the superblock and the number of blocks used by the list of inodes. The number of inodes must be less than 65,535. All other file system checks depend on the correctness of these sizes.

You may see the following message when *fsck* detects an inconsistency in the file system size.

Size check: *fsize x isize y*

More blocks are used for the inode list *y* than there are blocks in the file system *x*, or there are more than 65,535 inodes in the file system. The *fsck* program ignores this file system and continues checking the next file system.

5.1.6 Scratch file errors

A scratch file is a temporary file that *fsck* creates when you invoke *fsck* with the *t* flag option. This option might be necessary if there is not enough core memory to hold *fsck*'s tables. You may see this message if *fsck* cannot create its own scratch file for a particular file system.

can't create *f*

This message means that *fsck*'s request to create a scratch file *f* failed. It ignores this file system and continues checking the next file system given. Check access modes of *f*.

5.1.7 Interactive messages

These messages require your yes or no response to the CONTINUE? prompt. "Yes" may be Y or y, and "no" may be N or n; these are shown in uppercase here. These messages may indicate a serious problem because the file system cannot be completely checked. You

may need to obtain additional technical assistance.

CANNOT SEEK: BLK *b*
CONTINUE?

A request to move to a specified block number *b* in the file system failed.

Possible responses to the CONTINUE? prompt are

Y Attempt to continue to run the file system check. If the problem persists, run *fsck* a second time to recheck this file system. If the block *b* was part of the virtual memory buffer cache, *fsck* will terminate with the message Fatal I/O error.

N Terminate *fsck*.

CANNOT READ: BLK *b*
CONTINUE?

The *fsck* program request for reading a specified block number *b* in the file system failed.

Possible responses to the CONTINUE? prompt are:

Y Attempt to continue to run the file system check. Try a second run of *fsck*. If the block *b* was part of the virtual memory buffer cache, *fsck* terminates with the message Fatal I/O error.

N Terminate *fsck*.

CANNOT WRITE: BLK *b*
CONTINUE?

The *fsck* program request for writing a specified block number *b* in the file system failed. The disk is write-protected.

Possible responses to the CONTINUE? prompt are:

Y Attempt to continue to run the file system check. A second run of *fsck* should be made to recheck this file system. If

the block *b* was part of the virtual memory buffer cache,
fsck terminates with the message Fatal I/O error.

N Terminate *fsck*.

5.2 Phase 1: check blocks and sizes

During this phase of execution, *fsck* checks the inode list. In this phase, *fsck* may discover error conditions that result from checking inode types, setting up the zero-link-count table, checking inode block numbers for bad or duplicate blocks, checking inode size, and checking inode format.

Although each individual inode has a small chance of becoming inconsistent, there are so many of them that it's almost as likely that an inconsistency will occur in the inode list as in the superblock.

The list of inodes is checked sequentially, starting with inode 1 (there is no inode 0) and going to the last inode in the file system. Each inode can be checked for inconsistencies involving format and type, link count, duplicate blocks, bad blocks, and inode size.

5.2.1 Inode type errors

Each inode contains a mode word that describes the type and state of the inode.

Inodes may be one of five types: regular, directory, special block, special character, or FIFO, according to the file involved. If an inode is not one of these types, it is of an illegal type. Tell *fsck* to clear the inode.

UNKNOWN FILE TYPE I=*i*
CLEAR?

The mode word of the inode *i* indicates that the inode is not a recognized A/UX file type. The problem is usually caused by a strange occurrence with the mode bits.

Possible responses to the CLEAR? prompt are:

- Y Deallocate inode *i* by zeroing its contents. This always invokes the UNALLOCATED error condition in phase 2 for each directory entry pointing to this inode.
- N Ignore this error condition.

5.2.2 Zero-link-count table errors

Each inode contains a stored count of the total number of directory entries linked to the inode. The `fsck` program verifies the link count of each inode by traversing down the total directory structure, starting from the root directory, and calculating an actual link count for each inode.

If the stored link count is nonzero and the actual link count is zero, no directory entry appears for the inode. If the stored and actual link counts are nonzero and unequal, a directory entry may have been added or removed without the inode being updated.

If the stored link count is nonzero and the actual link count is zero, `fsck` can link the disconnected file to the `lost+found` directory (at your direction). If the stored and actual link counts are nonzero and unequal, `fsck` can replace the stored link count by the actual link count.

LINK COUNT TABLE OVERFLOW CONTINUE?

An internal table for `fsck` has no more room. This is a rare error. Contact your Apple representative for assistance.

Possible responses to the CONTINUE? prompt are:

- Y Continue with the program. This error condition makes a complete check of the file system impossible. You should make a second run of `fsck` to recheck this file system. If another allocated inode with a zero link count is found, this error condition is repeated.
- N Terminate `fsck`.

5.2.3 Bad or duplicate blocks

Each inode contains a list of pointers to lists (indirect blocks) of all the blocks claimed by the inode.

The `fsck` program checks each block number claimed by an inode for a value lower than that of the first data block or greater than that of the last block in the file system. If the block number is outside this range, it is a bad block number. If an indirect block was not written to disk, an inode may contain many bad blocks. In this case, `fsck` clears both inodes (at your direction).

The `fsck` program compares each block number claimed by an inode to a list of already allocated blocks. If a block number is already claimed by another inode, the block number is added to a list of duplicate blocks. Otherwise, the list of allocated blocks is updated to include the block number. If there are any duplicate blocks, `fsck` makes a partial second pass of the inode list to find the inode of the duplicated block. This is necessary because without examining the files associated with these inodes for correct content, `fsck` does not have enough information to decide which inode is corrupted and should be cleared. Usually, the inode with the earliest modify time is incorrect and should be cleared. This condition may be the result of a file system containing blocks claimed by both the free list and other parts of the file system.

A large number of duplicate blocks in an inode may be due to an indirect block not being written to disk. In this case, `fsck` clears both inodes (at your direction).

`b BAD I=i`

The variable *i* stands for the actual number on your screen. Inode *i* contains block number *b*, which is lower than the number of the first data block in the file system or greater than the number of the last block in the file system.

This error condition may invoke the EXCESSIVE BAD BLKS error condition in phase 1 if inode *i* has too many block numbers outside the file system range.

This error condition always invokes the BAD/DUP error condition in phase 2 and phase 4.

EXCESSIVE BAD BLKS I=*i*
CONTINUE?

As before, *i* stands for the actual number on your screen. The number of blocks with a number lower than the number of the first data block in the file system or greater than the number of the last block in the file system associated with inode *i* is too high to be acceptable. Usually, 10 is the cutoff point.

Possible responses to the CONTINUE? prompt are:

Y Ignore the rest of the blocks in this inode and continue checking with next inode in the file system. This error condition makes a complete check of the file system impossible. A second run of `fsck` should be made to recheck this file system.

N Terminate `fsck`.

b DUP I=*i*

Inode *i* contains block number *b*, which is already claimed by another inode.

This error condition may invoke the EXCESSIVE DUP BLKS error condition in phase 1 if inode *i* has too many block numbers claimed by other inodes.

This error condition always invokes phase 1B and the BAD/DUP error condition in phase 2 and phase 4.

EXCESSIVE DUP BLKS I=*i*
CONTINUE?

As before, *i* stands for the actual number on your screen. The number of blocks claimed by other inodes is too high to be acceptable. Usually, 10 is the cutoff point.

Possible responses to the CONTINUE? prompt are:

Y Ignore the rest of the blocks in this inode and continue checking with the next inode in the file system. This error condition will not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system.

N Terminate `fsck`.

DUP TABLE OVERFLOW
CONTINUE?

An internal table in `fsck`, containing duplicate block numbers, has no more room. Possible responses to the `CONTINUE?` prompt are:

- Y Continue with file system check. This error condition makes a complete check of the file system impossible. Try a second run of `fsck`. If another duplicate block is found, this error condition repeats.
- N Terminate `fsck`.

5.2.4 Inode size errors

Each inode contains a 32-bit (4-byte) field. This size indicates the number of characters in the file associated with the inode. This size can be checked for inconsistencies; for example, directory sizes are not a multiple of 16 characters, or the number of blocks actually used does not match the number indicated by the inode size.

A directory inode within the file system has the directory bit on in the inode mode word. The directory size must be a multiple of 16 because a directory entry contains 16 bytes (2 bytes for the inode number and 14 bytes for the file or directory name).

The `fsck` program warns of such directory misalignment. This is only a warning because not enough information can be gathered to correct the misalignment.

A rough check of an inode's size field consistency can be performed by computing from the size field the number of blocks that should be associated with the inode and comparing that number to the actual number of blocks claimed by the inode.

The `fsck` program calculates the number of blocks by dividing the number of characters in an inode by the number of characters per block and rounding up. The `fsck` program adds one block for each indirect block associated with the inode. If the actual number of blocks does not match the computed number of blocks, `fsck` warns of a possible file-size error. This is only a warning because the system does not fill in blocks in files created in random order.

Note: These messages are only warnings. If you use the `q` option on the `fsck` command line, these messages are not printed.

POSSIBLE FILE SIZE ERROR I=*i*

Again, *i* stands for the actual number on your screen. The inode size does not match the actual number of blocks used by the inode. If this error occurs, write down the inode number. When `fsck` finishes, continue in single-user mode, mount the file system in which the error occurred, and get its corresponding filename in the following way: Enter the command

```
ncheck -i i fs
```

where *i* is the number of the inode as provided by the POSSIBLE FILE SIZE ERROR message, and *fs* stands for the name of the file system in which the message occurred. *Be sure to use the full pathname of the device, for example /dev/dsk/c2d0s0, and not the mount point of the file system.* The `ncheck` command prints the name of the file on the screen. Copy it into a temporary name and run `sync`. Examine the file and, if you want to retain it, remove the original and give the temporary file its original name. Note that just using `mv` would not do the job.

DIRECTORY MISALIGNED I=*i*

Again, *i* stands for the actual number on your screen. The size of a directory inode is not a multiple of the size of a directory entry (usually 16). It should be cleared.

5.2.5 Inode format errors

Each inode contains a mode word that describes the type and state of the inode. Inodes may be found in one of three states: unallocated, allocated, and neither unallocated nor allocated. This last state indicates an incorrectly formatted inode. An inode can get in this state if bad data is written into the inode list, a possible result of a hardware failure. The only corrective action is for `fsck` to clear the inode.

PARTIALLY ALLOCATED INODE I=*i*
CLEAR?

Inode *i* is neither allocated nor unallocated.

Possible responses to the CLEAR? prompt are:

Y Deallocate inode *i* by zeroing its contents.

N Ignore this error condition.

5.3 Phase 1B: rescan for more duplicates

When a duplicate block is found in the file system, the file system is rescanned to find the inode that previously claimed that block. If the duplicate block is found, the following error condition occurs.

b DUP *I=i*

Inode *i* contains block number *b*, which is already claimed by another inode.

This error condition always invokes the DUPS/BAD error condition in phase 2.

Inodes with overlapping blocks may be determined by examining this error condition and the DUP error condition in phase 1.

5.4 Phase 2: check pathnames

This phase removes directory entries pointing to inodes that had error conditions from phase 1 and phase 1B. In this phase, *fsck* may discover error conditions that result from root inode mode and status, directory inode pointers out of range, and directory entries pointing to bad inodes.

5.4.1 Root inode mode and status errors

ROOT INODE UNALLOCATED. TERMINATING

The root inode (always inode number 2) has no allocated mode bits. This error condition indicates a serious problem that you may not be able to solve without technical assistance. The program will terminate.

ROOT INODE NOT DIRECTORY
FIX?

The root inode (usually inode number 2) is not of the directory type.

Possible responses to **FIX?** prompt are:

- Y** Make the root inode a directory. If the root inode's data blocks are not directory blocks, a very large number of error conditions will result.
- N** Terminate **fsck**.

DUPS/BAD IN ROOT INODE
CONTINUE?

During phase 1 or phase 1B, **fsck** found duplicate blocks or bad blocks in the root inode (usually inode number 2) for the file system.

Possible responses to the **CONTINUE?** prompt are

- Y** Ignore **DUPS/BAD** error condition in the root inode and attempt to continue to run the file system check. If the root inode is not correct, then a large number of other error conditions may result.
- N** Terminate **fsck**.

5.4.2 Directory inode pointers range errors

I OUT OF RANGE I=i NAME=f
REMOVE?

A directory entry *f* has an inode number *i* greater than the end of the inode list.

Possible responses to the **REMOVE?** prompt are:

- Y** Remove the directory entry *f*.
- N** Ignore this error condition.

5.4.3 Directory entries pointing to bad inodes

UNALLOCATED I=i OWNER=o MODE=m SIZE=s MTIME=t NAME=f
REMOVE?

A directory entry *f* has an inode *i* without allocated mode bits. The owner *o*, mode *m*, size *s*, modify time *t*, and filename *f* are printed.

If the file system is not mounted and the `n` option wasn't specified, the entry is removed automatically if the inode it points to contains size 0.

Possible responses to REMOVE? prompt are:

Y Remove the directory entry *f*.

N Ignore this error condition.

DUPS/BAD I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t* DIR=*f*
REMOVE?

During phase 1 or phase 1B, `fsck` has found duplicate blocks or bad blocks associated with directory entry *f*, directory inode *i*. The owner *o*, mode *m*, size *s*, modify time *t*, and directory name *f* are printed.

Possible responses to REMOVE? prompt are:

Y Remove the directory entry *f*.

N Ignore this error condition.

DUPS/BAD I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t* FILE=*f*
REMOVE?

During phase 1 or phase 1B, `fsck` has found duplicate blocks or bad blocks associated with directory entry *f*, inode *i*. The owner *o*, mode *m*, size *s*, modify time *t*, and filename *f* are printed.

Possible responses to the REMOVE? prompt are:

Y Remove the directory entry *f*.

N Ignore this error condition.

BAD BLK *b* IN DIR I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t*

This message occurs only when the `q` option is used. It means that a bad block was found in inode *i*.

The `fsck` program checks directory blocks for nonzero padded entries, inconsistent . and . . entries, and embedded slashes in the name field. This error message indicates that, at a later time, you should either remove the directory inode if the entire block looks bad or change (or remove) those directory entries that look bad.

5.5 Phase 3: check connectivity

During phase 3, `fsck` checks the directory connectivity seen in phase 2. In this phase, `fsck` may discover error conditions that result from unreferenced directories and missing or full `lost+found` directories.

5.5.1 Unreferenced directories

UNREF DIR I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t*
RECONNECT?

The directory inode *i* was not connected to a directory entry when the file system was traversed. The owner *o*, mode *m*, size *s*, and modify time *t* of directory inode *i* are printed. The `fsck` forces the reconnection of a nonempty directory.

Possible responses to the RECONNECT? prompt are:

- Y Reconnect the directory inode *i* to the file system in the directory for lost files (usually `lost+found`). This may invoke `lost+found` error conditions (described later) if there are problems connecting directory inode *i* to `lost+found`. This may also invoke the CONNECTED message (described later) if the link was successful.
- N Ignore this error condition. This will always invoke the UNREF error condition (described later).

SORRY. NO `lost+found` DIRECTORY

There is no `lost+found` directory in the root directory of the file system. In this case, `fsck` ignores the request to link a directory in `lost+found`. This will always invoke the UNREF error condition (described later). If `lost+found` exists, check its access modes. See `fsck(1M)` and `mklost+found(1M)` in *A/UX System Administrator's Reference* for further details.

SORRY. NO SPACE IN `lost+found` DIRECTORY

There is no space in the `lost+found` directory to add another entry; `fsck` ignores the request to link a directory in `lost+found`. This will always invoke the UNREF error condition (described later). Clean out unnecessary entries in the `lost+found` directory or make it larger. See `fsck(1M)` in *A/UX System Administrator's Reference* for further details.

DIR I=*i* CONNECTED. PARENT WAS I=*j*

This is an advisory message indicating that a directory inode *i* was successfully connected to the `lost+found` directory. The parent inode *j* of the directory inode *i* is replaced by the inode number of the `lost+found` directory.

5.6 Phase 4: check reference counts

During phase 4, `fsck` checks the directory connectivity seen in phase 2 and phase 3.

In phase 4, `fsck` reports errors that result from unreferenced files; a missing or full `lost+found` directory; incorrect link counts for files, directories, or special files; unreferenced files and directories; bad and duplicate blocks in files and directories; and incorrect total free inode counts.

5.6.1 Unreferenced files

UNREF FILE I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t*
RECONNECT?

Inode *i* was not connected to a directory entry when the file system was traversed. The owner *o*, mode *m*, size *s*, and modify time *t* of inode *i* are printed. If the `Option` is not set and the file system is not mounted, empty files are not reconnected and are cleared automatically.

Possible responses to the RECONNECT? prompt are:

- Y Reconnect inode *i* to the file system in the directory for lost files (usually `lost+found`). This may invoke a `lost+found` error condition (described later) if there are problems connecting inode *i* to `lost+found`.
- N Ignore this error condition. This will always invoke the `CLEAR?` error condition (described later).

5.6.2 `lost+found` directory errors

SORRY. NO `lost+found` DIRECTORY

There is no `lost+found` directory in the root directory of the file system; `fsck` ignores the request to link a file in `lost+found`. This will always invoke `CLEAR?` error condition (described later).

Check the access modes of `lost+found`. Also see `mklost+found(1M)` in *A/UX System Administrator's Reference*.

SORRY. NO SPACE IN `lost+found` DIRECTORY

There is no space to add another entry to the `lost+found` directory in the root directory of the file system; `fsck` ignores the request to link the file. This will always invoke the `CLEAR?` error condition (described next). Check the size and contents of `lost+found`.

CLEAR?

The inode mentioned in the immediately preceding error condition cannot be reconnected.

Possible responses to the `CLEAR?` prompt are:

- Y** Deallocate the inode mentioned in the immediately preceding error condition by zeroing its contents.
- N** Ignore this error condition.

5.6.3 Incorrect free inode counts

The superblock contains a count of the total number of free inodes in the file system. The `fsck` program compares this count to the number of inodes it found free in the file system. If the counts do not agree, `fsck` may replace the count in the superblock with the actual free inode count.

**LINK COUNT FILE `I=i` OWNER=`o` MODE=`m` SIZE=`s`
MTIME=`t` COUNT=`x` SHOULD BE `y`
ADJUST?**

The link count for inode `i`, which is a file, is `x` but should be `y`. The owner `o`, mode `m`, size `s`, and modify time `t` are printed.

Possible responses to the `ADJUST?` prompt are:

- Y** Replace the link count of file inode `i` with `y`.
- N** Ignore this error condition.

LINK COUNT DIR I=*i* OWNER=*o* MODE=*m* SIZE=*s*
MTIME=*t* COUNT=*x* SHOULD BE *y*
ADJUST?

The link count for inode *i*, which is a directory, is *x* but should be *y*. The owner *o*, mode *m*, size *s*, and modify time *t* of directory inode *i* are printed.

Possible responses to the ADJUST? prompt are:

- Y Replace the link count of directory inode *i* with *y*.
- N Ignore this error condition.

LINK COUNT F I=*i* OWNER=*o* MODE=*m* SIZE=*s*
TIME=*t* COUNT=*x* SHOULD BE *y*
ADJUST?

The link count for file *f* inode *i* is *x* but should be *y*. The filename *f*, owner *o*, mode *m*, size *s*, and modify time *t* are printed.

Possible responses to the ADJUST? prompt are:

- Y Replace the link count of inode *i* with *y*.
- N Ignore this error condition.

5.6.4 Unreferenced files/directories

UNREF FILE I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t*
CLEAR?

Inode *i*, which is a file, was not connected to a directory entry when the file system was traversed. The owner *o*, mode *m*, size *s*, and modify time *t* of inode *i* are printed. If you don't set the *n* option and the file system is not mounted, empty files are cleared automatically.

Possible responses to CLEAR? prompt are:

- Y Deallocate inode *i* by zeroing its contents.
- N Ignore this error condition.

UNREF DIR I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t*
CLEAR?

Inode *i*, which is a directory, was not connected to a directory entry when the file system was traversed. The owner *o*, mode *m*, size *s*, and modify time *t* of inode *i* are printed. If you don't set the *-n* option and the file system is not mounted, empty directories are cleared automatically. Nonempty directories are not cleared.

Possible responses to the CLEAR? prompt are:

YES Deallocate inode *i* by zeroing its contents.

NO Ignore this error condition.

5.6.5 Bad and duplicate blocks in files and directories

BAD/DUP FILE I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t*
CLEAR?

During phase 1 or phase 1B, *fsck* has found duplicate blocks or bad blocks associated with file inode *i*. The owner *o*, mode *m*, size *s*, and modify time *t* of inode *i* are printed.

Possible responses to the CLEAR? prompt are:

Y Deallocate inode *i* by zeroing its contents.

N Ignore this error condition.

BAD/DUP DIR I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t*
CLEAR?

During phase 1 or phase 1B, *fsck* has found duplicate blocks or bad blocks associated with directory inode *i*. The owner *o*, mode *m*, size *s*, and modify time *t* of inode *i* are printed.

Possible responses to the CLEAR? prompt are:

Y Deallocate inode *i* by zeroing its contents.

N Ignore this error condition.

FREE INODE COUNT WRONG IN SUPERBLK
FIX?

The actual count of the free inodes doesn't match the count in the superblock of the file system. If you specify the *q* option, the count is fixed automatically in the superblock.

Possible responses to the `FIX?` prompt are:

- `Y` Replace the count in superblock by the actual count.
- `N` Ignore this error condition.

5.7 Phase 5: check free list

The free list starts in the superblock and continues through the free list link blocks of the file system.

Each free list link block can be checked for a list count out of range, for block numbers out of range, and for blocks already allocated within the file system. A check is made to see that all the blocks in the file system were found.

The free list check begins in the superblock. The `fsck` program checks the list count for a value of less than 0 or greater than 50. It also checks each block number for a value of less than the first data block in the file system or greater than the last block in the file system. Then it compares each block number to a list of already allocated blocks. If the free list link block pointer is nonzero, `fsck` reads in the next free list link block and repeats the process.

When all the blocks have been accounted for, `fsck` checks whether the number of blocks used by the free list plus the number of blocks claimed by the inodes equal the total number of blocks in the file system.

If anything is wrong with the free list, `fsck` may rebuild the list, excluding all blocks in the list of allocated blocks.

The superblock also contains a count of the total number of free blocks in the file system. The `fsck` program compares this count to the number of blocks it found free in the file system. If the counts do not agree, `fsck` may replace the count in the superblock with the actual free block count.

During phase 5, `fsck` lists error conditions resulting from bad blocks in the free list, bad free-block count, duplicate blocks in the free list, unused blocks from the file system not in the free list, and incorrect total free-block count.

EXCESSIVE BAD BLKS IN FREE LIST
CONTINUE?

The free list contains more than a tolerable number (usually 10) of blocks with a value less than the first data block in the file system or greater than the last block in the file system.

Possible responses to the CONTINUE? prompt are:

Y Ignore the rest of the free list and continue execution of *fsck*. This error condition always invokes the BAD BLKS IN FREE LIST error condition in phase 5.

N Terminate *fsck*.

EXCESSIVE DUP BLKS IN FREE LIST
CONTINUE?

The free list contains more than a tolerable number (usually 10) of blocks claimed by inodes or earlier parts of the free list.

Possible responses to the CONTINUE? prompt are:

Y Ignore the rest of the free list and continue execution of *fsck*. This error condition always invokes the DUP BLKS IN FREE LIST error condition in phase 5.

N Terminate *fsck*.

BAD FREEBLK COUNT

The count of free blocks in a free list link block is greater than 50 or less than 0. This error condition always invokes the BAD FREE LIST condition in phase 5.

x BAD BLKS IN FREE LIST

The free list contains *x* blocks with a block number lower than the first data block in the file system or greater than the last block in the file system. This error condition always invokes the BAD FREE LIST condition in phase 5.

x DUP BLKS IN FREE LIST

The free list contains *x* blocks claimed by inodes or earlier parts of the free list. This error condition always invokes the BAD FREE LIST condition in phase 5.

***x* BLK(S) MISSING**

The free list does not contain *x* blocks unused by the file system. This error condition always invokes the BAD FREE LIST condition in phase 5.

**FREE BLK COUNT WRONG IN SUPERBLK
FIX?**

The actual count of free blocks does not match the count in the superblock of the file system.

Possible responses to the FIX? prompt are:

- Y** Replace the count in the superblock with the actual count.
- N** Ignore this error condition.

**BAD FREE LIST
SALVAGE?**

During phase 5, *fsck* has found bad blocks in the free list, duplicate blocks in the free list, or blocks missing from the file system. If the *q* option is specified, the free list is salvaged automatically.

Possible responses to the SALVAGE? prompt are:

- Y** Replace the actual free list with a new free list. The new free list is ordered to shorten the time spent waiting for the disk to rotate into position.
- N** Ignore this error condition.

5.8 Phase 6: salvage free list

This phase concerns itself with the free list reconstruction. During this phase, *fsck* lists error conditions resulting from the blocks-to-skip and blocks-per-cylinder values.

Default free list spacing assumed

This is an advisory message indicating the blocks-to-skip value is greater than the blocks-per-cylinder value, the blocks-to-skip value is less than 1, the blocks-per-cylinder value is less than 1, or the blocks-per-cylinder value is greater than 500. The default values of 9 blocks-to-skip and 400 blocks-per-cylinder are used. See `fsck(1M)` in *A/UX System Administrator's Reference* for further details.

5.9 Cleanup

Once a file system has been checked, a few cleanup functions are performed. The `fsck` program lists advisory messages about the file system and its modify status.

`x files y blocks z free`

This is an advisory message indicating that the file system checked contained `x` files using `y` blocks leaving `z` blocks free in the file system.

***** Fixed root filesystem, rebooting A/UX! *****

This message indicates that `fsck` has modified the root file system to fix inconsistencies it found. The `fsck` program forces a reboot because it can fix the file system image only on the disk but not in memory. Since A/UX periodically issues a `sync(1)` call to update the file system from memory, the forced reboot of the system prevents the file system repair from being undone by the automatic `sync` call.

***** FILE SYSTEM WAS MODIFIED *****

This advisory message indicates that `fsck` has modified a non-root file system to fix the inconsistencies it found. A/UX continues to run.

PLACE TAB
HERE

SYSTEM
ACCOUNTING

BACK OF TAB

Chapter 9

System Accounting Package

Contents

1. Introduction	1
2. Routine accounting procedures	1
2.1 The <code>cron</code> file	3
2.2 Updating holidays	4
2.3 Daily operation	5
2.3.1 The <code>ckpacct</code> procedure	6
2.3.2 The <code>dodisk</code> procedure	6
2.3.3 The <code>chargefee</code> procedure	7
2.3.4 The <code>runacct</code> procedure	7
2.3.5 The <code>prdaily</code> procedure	12
2.4 Restarting <code>runacct</code>	12
2.4.1 In case <code>runacct</code> fails	13
2.4.2 Error messages	15
2.4.3 Fixing corrupted files	17
2.4.4 Fixing <code>wtmp</code> errors	17
2.4.5 Fixing <code>tacct</code> errors	17
2.4.6 The <code>monacct</code> procedure	18
3. Special accounting procedures: <code>acctcom</code>	18
3.1 The <code>acctcom</code> command	19

BLANK PAGE

Chapter 9

System Accounting Package

1. Introduction

Two packages provided with your A/UX system allow you to keep track of the details of system operation and usage:

- The system accounting package collects information and generates reports on buffer activity, CPU utilization in general, device activity, and so on. This chapter discusses the system accounting package.
- The system activity package permits you to keep track of all low-level activity in your system. For information on the system activity package, see Chapter 10, "System Activity Package."

The system accounting package collects detailed information on system usage and allows you to generate a comprehensive system usage report on a daily and monthly basis. (Note that the report provides information on overall system usage, not on individual users.) This report function, for the most part, is automatic and is described in the next section, "Routine Accounting Procedures." You can also produce customized reports by issuing accounting commands; these are described in "Special Accounting Procedures: acctcom."

2. Routine accounting procedures

The system uses routine accounting procedures to generate information describing what a user is doing and how often the user invokes a specific command. It also generates information on overall system usage, frequency of usage, and system resource allocation.

Take these steps to turn on system accounting and automate its operation:

1. Log in as the root user.

2. Make sure the following lines are in the `/etc/rc` file and are not commented out (if they are preceded by a number sign (#), remove it):

```
/bin/su adm -c /usr/lib/acct/startup
echo process accounting started
```

The first line is a command that activates the accounting system when the system is brought up. The second line prints "process accounting started" when the accounting system starts up.

3. Make sure that the following lines in the `/usr/spool/cron/crontabs/adm` file are not commented out (if they are preceded by a number sign (#), remove it):

```
0 4 * * 1-6 /usr/lib/acct/runacct
                2> /usr/adm/acct/nite/fd2log
0 2 * * 4 /usr/lib/acct/dodisk
5 * * * * /usr/lib/acct/ckpacct
15 5 1 * * /usr/lib/acct/monacct
0 * * * 0,6 /usr/lib/sa/sa1
0 18-7 * * 1-5 /usr/lib/sa/sa1
0 8-17 * * 1-5 /usr/lib/sa/sa1 1200 3
0 20 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e
                18:00 -i 3600 -uybd
```

Note: The first two lines and the last two lines in the example *must* appear as one line in the `adm` file. These lines are broken here only because the book page does not accommodate the long measure.

The `adm` file instructs `cron` to run the daily accounting automatically. The line

```
15 5 1 * * /usr/lib/acct/monacct
```

uses the `monacct` file to clean up all daily reports and daily total accounting files. The `adm` file also deposits one monthly total report and one monthly total accounting file in the `fiscal` directory after `runacct` has completed the last day's entries.

By default, `monacct` uses the current month's date as the suffix for the filenames.

4. Make sure that the following line is in the `/usr/adm/.profile` file:

```
PATH=/usr/lib/acct:/bin:/usr/bin
```

This ensures that the operating system has access to all the necessary files and scripts to process the accounting system automatically.

2.1 The cron file

Although you don't need to understand how the `cron` file works to use the accounting package, knowing how the entries are written and what they do makes it easy to modify entries if you need to. This section is a quick introduction to `cron`.

The `cron` program performs specified functions at specified times. These functions and times are specified in the file `/usr/spool/cron/crontabs/adm`, which is read and acted on by `cron`. Each line of text in this file is a small script consisting of six fields that together specify a process to be executed at a certain time. The fields are separated by spaces or tabs. The first five fields specify a time. They specify, in order,

<i>minute</i>	Time in minutes (0-59)
<i>hour</i>	Time in hours (0-23)
<i>day-of-month</i>	Day of the month (1-31)
<i>month-of-year</i>	Month of the year (1-12)
<i>day-of-week</i>	Day of the week (0-6, with 0 as Sunday)

Each of these may be either an asterisk (which means ignore all cases) or a list of elements separated by commas (which specify specific cases to be activated). An element may be one number or two numbers separated by a hyphen (meaning an inclusive range).

You specify days in two fields: *day-of-month* and *day-of-week*. If you specify both, both are used. To specify days using only one field, set the other to `*`.

The sixth field is the process to be executed. A percent sign in this field (unless masked by \) is translated as the signal for a new line. The shell executes only the first line (up to the % or the end of the line).

In this example, the line

```
15 5 1 * * /usr/lib/acct/monacct
```

means this: at the 15th minute of the fifth hour on the first day of each month, run the monacct program, which is located in the /usr/lib/acct directory. The asterisks in the *month-of-year* and the *day-of-week* fields tell the system to ignore these fields.

You can change the field entries in the file /usr/spool/cron/crontabs/adm and run the crontab command on it, for instance, to run the accounting procedures more frequently or even print a weekly report instead of a monthly one.

2.2 Updating holidays

The file /usr/lib/acct/holidays contains the prime/nonprime table, which gives the start of prime time and the start of nonprime time for your operating system, using a 24-hour system (0100 to 2400 hours). Note that the hour 2400 automatically converts to 0000. Information on changing the prime/nonprime table to reflect individual preferences is given in this section. For example, during normal business operation, prime time is considered to be from 8:30 A.M. to 5:30 P.M. The table also contains the holiday schedule for the year, which you can adjust to include additional holidays.

The format of the holidays file includes:

Comment lines

Comment lines can appear anywhere in the file but they *must* be preceded by an asterisk so that the system won't read them.

Year designation line

The year designation line *must* be the first data line in the file and can appear only once in the file. It consists of three fields of four digits each:

```
yyyy hhmm hhmm (number of spaces irrelevant)
```

- The first field is the current year. Be sure to set it correctly, because an incorrect year entry affects the holiday entries.

- The second field is the start of prime time, in 24-hour time. Prime time refers to the peak activity period for your system. In most businesses, for example, prime time starts at 8:30 A.M. (0830 in a 24-hour system) to coincide with the start of the business day.
- The third field is the start of nonprime time, in 24-hour time. This field represents the end of peak activity for the operating system, usually 5:30 P.M. (or 1730).
- Your system may be set to start prime time at 0800 and nonprime time at 1700 (5:00 P.M.). If your company begins work at 8:00 P.M., for example, you may want to change the start of prime time to 2000 to reflect this. You would also change the nonprime time to reflect the close of business, say 2:00 A.M., as 0200.

Note: The hour 2400 automatically converts to 0000.

Company holiday lines

You can make entries for national and local holidays on the line following the year designation line. The format is

day-of-year month day description-of-holiday

The *day-of-year* is a number from 1 to 366, indicating the day for the corresponding holiday. The other three fields are not used by A/UX and are provided for commentary only.

Note: Remember to separate all entries with tabs and not with spaces.

2.3 Daily operation

The lines of text you enter in the `/usr/spool/cron/crontabs` file cause the system to automatically run the `startup`, `ckpacct`, `turnacct`, `dodisk`, `runacct`, and `monacct` procedures. Each of these six procedures is described in this section. Only some of the procedures are available. Other procedures—including `acctcom`—are also described. If you add these others to the `/usr/spool/cron/crontabs/adm` file and run the `crontab` command on the edited file, they are processed automatically. For

example, `acctcom` is discussed in "Special Accounting Procedures: `acctcom`."

When you bring the system into multi-user mode, `/usr/lib/acct/startup` is executed. This program has three effects:

- The `acctwtmp` program records a boot in the `/etc/wtmp` file. It uses your system name as the login name in the file.
- `Turnacct` begins the process accounting. The `accton` program is executed, and the collected data is stored in the `/usr/adm/pacct` file. This file is later read by the reporting function and summarized in the daily and monthly reports.
- The remove shell procedure is executed. This procedure cleans up the saved `pacct` and `wtmp` files that `runacct` creates.

2.3.1 The `ckpacct` procedure

After process accounting has begun, `cron` executes the `ckpacct` procedure every hour. This procedure checks the size of the `pacct` file. The `ckpacct` procedure begins the process of creating multiple `pacct` files when the file grows to 1000 blocks. It executes the `turnacct` command with the `switch` option (which turns the process accounting off), moves the current `/usr/adm/pacct` data to `/usr/adm/pacct/incr` (where `incr` is a number that starts with 1 and increases by one for each additional `pacct` file `turnacct` creates), and then turns the process accounting back on. This limits the `pacct` files to a reasonable size. If you ever need to restart `runacct`, the smaller file size makes the job easier.

2.3.2 The `dodisk` procedure

The `cron` program invokes the `dodisk` procedure to perform the disk accounting functions. The command is structured as follows:

```
/usr/lib/acct/dodisk [-o][file1...]
```

If you specify no options (the default), the procedure performs disk accounting on the files in `/etc/fstab` and `/etc/inittab`, which is a list of all file systems in the disk partition.

If you use the `-o` flag, a slower version of disk accounting by login directory is done.

The *file1* specifies the name(s) of the file system(s) where the disk accounting is done. If you use the *files* argument, disk accounting will be done on these file systems only. If you use the *-o* flag, *files* should be the names of the directories on which the file systems are mounted. If you omit the *-o* flag, *files* should be the special filenames of mountable file systems.

2.3.3 The chargefee procedure

You can invoke the chargefee shell procedure to charge a number of units to a login name. The command syntax is

```
/usr/lib/acct/chargefee login-name number
```

For example, you charge login name john for two units (\$2.00) for his system usage. This information is then written to */usr/adm/fee* and merged with the other accounting records during the night. This information then appears in the FEE column in the daily report generated by *runacct*.

2.3.4 The runacct procedure

The main daily accounting procedure is *runacct*. This section covers the *runacct* command itself, the error messages it generates, and the way to recover if the procedure fails.

The *runacct* command has the following form:

```
/usr/lib/acct/runacct [mmdd][mmdd state]
```

You can use the options to restart *runacct* after a failure. They are explained later in the section.

The entries made to the */usr/spool/cron/crontabs/adm* file initiate the *runacct* procedure during nonprime hours. The *runacct* procedure automatically processes the connect, fee, disk, and process accounting files and prepares daily and cumulative summary files, which are then read by *prdaily* or used for billing purposes. When you run *monacct*, these daily reports are summarized into a monthly report.

To read the daily report, type

```
/usr/lib/acct/prdaily | more
```

The prdaily reporting function is covered in more detail later in this chapter. The preceding command prints a report generated by the runacct procedure. The report consists of a header and five parts.

The header displays the dates of the current reporting period, for example:

```
Oct 26 10:04 1987  DAILY REPORT FOR A/UX Page 1

from Tue Oct 20 04:00:13 1987
to   Tue Oct 20 04:00:13 1987
```

Following the date is a listing of the /etc/wtmp entries generated by the acctwtmp program. This listing includes any reboots, shutdowns, power failure recoveries, date changes, and so on, that occurred during the reporting period, for example,

```
2  date changes
```

Commands used to produce the report are displayed, for example,

```
1  runacct
1  acctconl
```

The first part of the report describes the connect accounting information on terminal usage, the number of sessions (logins, logouts), and the amount of time each terminal was used. The fields in this part of the report are as follows:

TOTAL DURATION

The amount of time the system was in multi-user mode.

LINE

The terminal line or access port used.

MINUTES

The amount of time the line was in use during the reporting period.

PERCENT

The value of MINUTES divided by TOTAL DURATION.

SESS, # ON

These columns give the number of logins on the line during the reporting period. This report is helpful in finding which lines

have been logged on, but not off.

OFF

Not only the number of logoffs, but also the number of interrupts on the line. If the # OFF exceeds the # ON by a large factor, it is possible that there is a bad connection or that the multiplexer, modem, or cable is going bad. You should monitor the /etc/wtmp file. If it grows rapidly, execute acctcon1 to see which TTY line is the noisiest. A large number of interrupts can affect the system adversely.

The next part of the report is a breakdown of system resource utilization by user. It does not give specific information on what each user was doing but provides information on the CPU time and connect time for each user. To charge users for system usage, see the information in the FEE column and the "The chargefee Procedure," earlier in this chapter.

UID The user ID (uid) for each login. For more information on uid, see Chapter 3, "User and Group Administration."

LOGIN NAME

The actual user login name. This column lets you differentiate the activities of users with the same uid. The next column gives CPU usage. This figure is broken down into two amounts: prime-time and nonprime-time usage.

KCORE-MINS

A cumulative measure of the amount of memory a process uses. It is measured in kilobytes per minute and is broken down into prime-time and nonprime-time usage.

CONNECT (MINS)

The amount of time a user was logged in. It is measured in "real time" and is broken down into prime-time and nonprime-time usage. If this number is high, and the number in the # OF PROCS column is low, the user probably logs in first thing in the morning and then hardly uses the terminal.

DISK BLOCKS

An accounting of the disk usage by user as it is calculated by the acctdusg program. The # OF PROCS column gives the

number of processes used. A very high number might indicate a shell process gone wild. The # OF SESS column tells you how many times each user logged in.

DISK SAMPLES

The number of times the acctdusg ran to obtain the information in the DISK BLOCKS column.

FEE A record of disk charges to each user ID. This information is written to /usr/adm/fee and merged with the other accounting records during the night. It then appears in the FEE column in the daily report.

The next two parts of the report, the DAILY COMMAND SUMMARY and MONTHLY COMMAND SUMMARY, summarize command usage over the report period. The report period is specified in the *number* argument to the monacct command; if it is not specified, the default is the current month. These parts are identical in format. The daily report summarizes the command usage for the report period, and the monthly report summarizes the command usage from the beginning of the month through the current period. Entries may appear in the monthly command name column that do not appear in the daily report. These are commands that were used sometime during the current month but not during the current reporting period. You can fine-tune the system by making commonly used commands more accessible.

The columns and their output are defined as follows:

DAILY, MONTHLY, and TOTAL KCOREMIN

Both the DAILY and MONTHLY command summaries are sorted by the TOTAL KCOREMIN column. This provides the total amount of memory a process uses per minute, measured in kilobytes.

COMMAND NAME

Self-explanatory. All shell commands, however, are lumped under the entry sh. For example, the command cd won't appear in the report; it is included in the sh column.

It is important to note that entries such as a.out, core, or mysterious command names indicate errors in compiled programs. If a compiled program is not named, it appears in the

report under the default name a.out. The name core indicates errors in the execution of a compiled program. You can use acctcom to tell you who used a suspicious command, perhaps an alias, or who has been exercising superuser privileges. See "Special Accounting Procedures," later in this chapter, for more information.

NUMBER CMNDS

Total number of times a command was executed.

TOTAL CPU-MIN

Total processing time dedicated to a command.

TOTAL REAL-MIN

Time it takes to process the command in real time, including the real-time usage of background processes as well.

MEAN SIZE-K

Calculated as $\text{TOTAL KCOREMIN} / \text{NUMBER CMNDS}$.

MEAN CPU-MIN

Calculated as $\text{NUMBER CMNDS} / \text{TOTAL CPU-MIN}$ used to execute the commands.

HOG FACTOR

The ratio of system availability to system utilization. It is calculated by dividing the total CPU time by the elapsed time. It provides a relative measure of the CPU time the process used during its execution.

CHARS TRNSFD

The total number of characters transferred by the read and write system calls. The number of characters is calculated command by command. It can be a negative number; the reads may outnumber the writes, for example.

BLOCKS READ

A total count of the physical block reads and writes that a process performs.

The last part of the accounting report is a compilation of all the logins on the system and the last date they logged in. The report looks like this:

Sep 29 04:05 1986 LAST LOGIN Page 1

```
86-09-25 apple
86-09-27 alice
00-00-00 phil
00-00-00 sys
86-09-29 john
```

The first column gives the date of the last login in *yy-mm-dd* format. The second column gives the login name itself. As you can see from the example, the date information can be blank. If the system is shut down for any reason, the last login date for all users who have not logged on since the shutdown is 00-00-00. You can use this part of the report to determine which users are no longer active. These users (excluding those who may have logged on prior to a crash, but not since) may be candidates for removal.

Of course, if your system date is set incorrectly, or the battery dies, all the information is potentially wrong.

2.3.5 The *prdaily* procedure

The script *prdaily* generates a printout of the *runacct* process. The report resides in */usr/adm/acct/sum/rprtmmdd*. The command syntax is

```
/usr/lib/acct/prdaily [-l][-c][mmdd]
```

The notation *mmdd* indicates the month and day of the report. You can generate previous daily reports using this option, specifying the month and day of the data you wish to see. Note that the report generated on 0611, for example, is actually the report on usage for 0610. Remember, the daily information is no longer available after *monacct* is run. The *-l* flag prints a report of exceptional usage by login identification for a date specified with *mmdd*. The *-c* flag prints a report of exceptional resource usage by command. You can use it on the current day's accounting data only. These values are considered to signal exceptional usage: CPU > 80, KCORE > 500, and CONNECT > 120.

2.4 Restarting *runacct*

The *runacct* procedure is designed to recognize possible errors and give warnings before terminating the process. During processing,

messages are written in the `/usr/adm/acct/nite/active` file to inform the operator of successful completion of the various phases of the procedure.

Diagnostics are written into the `fd2log` (all `runacct` files are located in the `/usr/adm/acct/nite` directory unless otherwise specified). The `runacct` procedure informs you if `lock` or `lock1` exists. To prevent generation of more than one report per day, the `lastday` file keeps a record of the month and day the program was last run.

The `runacct` procedure does not damage active accounting or summary files. It records its progress by writing messages into the file `/usr/adm/acct/nite/active`. When it detects an error, it writes a message to the console, sends mail to `root` and `adm`, and then terminates.

The `runacct` procedure uses a series of lock files to prevent reinvocation of the accounting process until the errors have been corrected. It uses the files `lock` and `lock1` to prevent simultaneous invocation; the file `lastdate` prevents more than one invocation per day.

2.4.1 In case `runacct` fails

If you must restart `runacct` after a failure, begin by following these steps:

1. Check for diagnostic error messages in the `activemmd` file located in the `/usr/adm/acct/nite` directory. If this file contains error messages and the lock files exist, check the `fd2log` for unusual messages.
2. Fix up any corrupted data files, such as `pacct` or `wtmp`.
3. Remove the `lock`, `lock1`, and `lastdate` files if they are present.

If `runacct` cannot complete the procedure for any reason (lock file encountered, error encountered, or the like), a message is written to the console (with copies sent via mail to `root` and `adm`), locks are removed, diagnostic files are saved, and the process is terminated. If you review the messages in the active file and at the console or in mail, you can determine at which point the process was stopped and why.

You can then restart the process at the appropriate location and let it finish.

To make it easier to recover from errors, `runacct` is broken down into separate, restartable states. Under ordinary circumstances, the state's name is written into `statefile` as each state is completed. The `runacct` procedure then checks `statefile` to determine what has been done and what state to process next. States are executed in the following order:

SETUP

Moves active accounting files into working files.

WTMPFIX

Verifies the integrity of the `wtmp` file and corrects date changes if necessary.

CONNECT1

Produces connect session records in `ctmp.h` format.

CONNECT2

Converts `ctmp.h` format files into `tacct.h` format.

PROCESS

Converts process accounting records into `tacct.h` format.

MERGE

Merges the connect and process accounting records.

FEES

Converts the output of `chargefee` into `tacct.h` format and merges it with the connect and process accounting records.

DISK

Merges the disk accounting records with connect, process, and fee accounting records.

MERGETACCT

Merges the daily total accounting records in `daytacct` with the summary total accounting records in `/usr/adm/acct/sum/tacct`.

CMS

Produces the command summaries.

USEREXIT

You can include customized accounting procedures here.

CLEANUP

Cleans up the temporary files and exits. COMPLETE appears in this file when runacct is finished.

The runacct procedure begins processing with the next state in statefile. If you want to begin processing at another state, include the desired state on the command line to designate where processing should begin. You must also include the argument *mmdd*, specifying the month and day for which runacct should rerun the accounting process.

For example,

```
nohup runacct 0601 2>>/usr/adm/acct/nite/fd2log&
```

restarts the accounting process for June 1 (reporting data for May 31) at the next state in statefile. If you enter

```
nohup runacct 0601 MERGE 2 >>/usr/adm/acct/nite/fd2log&
```

runacct restarts on June 1 at the merge state.

Normally it is not a good idea to restart runacct in the setup state. Instead, run SETUP manually and restart by giving the command

```
runacct mmdd WTMPFIX
```

If runacct failed in the process state, be sure to remove the last ptacct file because it will not be complete.

2.4.2 Error messages

The acctcms -a command produces a core file in /usr/adm/acct each day that system accounting runs.

The runacct program produces a core dump in /usr/adm/acct if given file names contain a period or an underscore.

If runacct is terminated, error messages are written into the activemmdd file in the /usr/adm/acct/nite directory. If this file and the lock files exist, check fd2log for unusual messages.

The following are some common error messages and possible solutions. The list is by no means complete.

ERROR: locks found, run aborted
The files lock and lock1 were found. You must remove them
to restart runacct.

ERROR: acctg already run for *date*: check
/usr/adm/acct/nite/lastdate
Today's date is the same as the last entry in lastdate; remove
the last entry in lastdate.

ERROR: turnacct switch returned rc= ?
Check the integrity of turnacct and accton.

Note: The accton program must be owned by root
and have the setuid bit set.

ERROR: Spacct?.*mmdd* already exists
File setups probably have already been run. Check the status of
files and run setups manually.

ERROR: /usr/adm/acct/nite/wtmp.*mmdd* already
exists. Run setups manually.
File setups have probably already been run. Check the status of
files and run setups manually.

ERROR: wtmpfix errors see
/usr/adm/act/nite/wtmperror
The wtmp file is corrupted. Use fwtmp to correct it.

ERROR: connect acctg failed: check
/usr/adm/acct/nite/log
The acctcon1 program encountered a bad wtmp file. Use
fwtmp to correct it.

ERROR: Invalid state, check
/usr/adm/acct/nite/active
The statefile is probably corrupted. Check it and read the
active file before restarting.

2.4.3 Fixing corrupted files

When it is necessary to restart `runacct`, you may need to recreate some of the files before proceeding. You can ignore some, and you can restore others from backups. Some files, however, *must* be fixed.

2.4.4 Fixing `wtmp` errors

If the date is changed while the system is in multi-user mode, a set of date change records is written into `/etc/wtmp`. The `wtmpfix` program is designed to modify the time stamps in the `wtmp` files when this happens. If there has been a combination of date changes and reboots, the `wtmpfix` program might not work, causing `acctconl` to fail.

If this happens, you should make the following adjustment:

```
cd /usr/adm/acct/nite
fwtmp < wtmp.mmdd > xwtmp
ed xwtmp
    delete corrupted records or
    delete all records from beginning up to date change
fwtmp -ic < xwtmp > wtmp.mmdd
```

If you can't fix the `wtmp` file, create a null `wtmp` file, which will prevent connect time from being charged incorrectly. The `actprcl` procedure is not able to determine which login used a specific process; it will charge the process to the first login in that user's password file.

2.4.5 Fixing `tacct` errors

If you are using the accounting system to charge users for system usage, you must maintain the integrity of the `tacct` file in the `/usr/adm/acct/sum` directory.

If `tacct` records have negative numbers, duplicate user IDs, or a user ID of 65,535, the file may be corrupted. First, check `sum/tacctprev` with `prtacct`. If it looks all right, patch up the latest `sum/tacct.mmdd`, then recreate `sum/tacct`.

A sample patchup follows:

```

cd /usr/adm/acct/sum
acctmerg -v < tacct.mmdd > xtacct
ed xtacct
    remove the bad records
    write duplicate uid records to another file
acctmerg -i < xtacct > tacct.mmdd
acctmerg tacctprev < tacct.mmdd > tacct

```

Note: You can recreate sum/tacct by merging all the tacct.mmdd files (the monacct procedure does this).

2.4.6 The monacct procedure

The monthly accounting summary is another automated procedure in the accounting package. You should invoke monacct once each month or once each accounting period. The line in the cron file

```
15 5 1 * * /usr/lib/acct/monacct
```

causes monacct to be invoked once per month (see "The cron File," earlier in this chapter.)

When run from the command line, the form of the monacct command is

```
/usr/lib/acct/monacct [number]
```

where *number* indicates a month or period. You can specify the week (01-52), the month (01-12), or the fiscal period, such as quarters (01-04). If you don't specify an argument, monacct uses the current month as the default. The monacct procedure creates summary files in /usr/adm/acct/fiscal and restarts summary files in /usr/adm/acct/sum.

3. Special accounting procedures: acctcom

Besides the user information you can get from the automated procedures, additional information about users is available. For example, the automated procedure does not tell you who is doing what, or when. One way to gather this information is by implementing additional accounting commands supplied with your system.

One such command is the `acctcom` command, described in the next section.

3.1 The `acctcom` command

The `acctcom` command is the most useful accounting command supplied with your system. It reports what processes are associated with a particular terminal, user, or group of users.

The command syntax of `acctcom` is

```
acctcom [options][file]
```

The `acctcom` command reads a specified file, the standard input, or `/usr/adm/pacct`, and writes the selected records to the standard output. Each record represents the execution of one process.

If you don't specify a file, and standard input is associated with a terminal, `/usr/adm/pacct` is read. If this isn't the case, the standard input is read. If *file* arguments are given, they are read in the order given. Each individual file is read in chronological order by process completion time. The `/usr/adm/pacct` file is usually the current file to be examined. A busy system, however, may have several `pacct` files to be processed.

The output generated by this command is similar in format to the report generated by `runacct`. It includes the following column headings:

COMMAND NAME

The command name is preceded by a number sign (#) if the command was executed with superuser privileges. By using the `-n` option of the command, you can find out which users (selected with the `-u` option) are executing the commands.

USER NAME

The login name of the user.

TTY NAME

The terminal device associated with the process. If a process is not associated with a known terminal, a period (.) appears in this column.

START TIME

The time the process began.

END TIME

The time the process terminated.

REAL (SEC)

The elapsed real time the process took to complete.

CPU (SEC)

The elapsed CPU time the process took to complete.

MEAN SIZE (K)

The average amount of memory (in kilobytes) used by a process over the number of invocations of the process.

The following information appears in the output if certain options are used with `runacct`. The options are explained next.

STAT

The system exit status.

HOG FACTOR

Ratio of system availability to system utilization. It is calculated as total CPU time over elapsed time.

KCORE MIN

The amount of kilobyte segments of memory used by a process.

CPU FACTOR

A measurement of user time over system time plus user time.

CHARS TRNSFD

The number of characters transferred by the read and write commands.

BLOCKS READ

A total count of the physical block reads and writes that a process performed.

The `acctcom` command can be used with several options. Here is a list of options with a brief explanation of what each does. Try a few to find out which ones are best suited to your purposes. Pay particular attention to the `-u` option, which describes user usage of the system. For a full listing of all the options, see `acctcom(1M)` in *A/UX System Administrator's Reference*.

- a The main `acctcom` option. In addition to printing the column headings in the preceding list, it prints some average statistics about the processes selected at the end of the report.
- f Prints a report with columns showing the number of `fork/exec` flags and the system exit status.
- h Displays the fraction of total available CPU time consumed by the process during its execution in a column titled `HOG FACTOR`.
- l This option is not extremely useful, given the state of tty information. You specify the option, and you always get information for all terminals.
- u Gives you a report of all system usage by a particular login name. The option requires the argument *user*, which specifies the login name about which you wish to generate a report. You can use it in conjunction with the `-s`, `-e`, `-S`, and `-E` options to limit the search to a specific time period. If you specify an incorrect login name, `-u` generates an error message and then produces the entire report anyway.
- g Similar to the `-u` option. Instead of printing system usage by user, however, it prints usage by the group. It requires the argument *group*, which may be either the group name or group ID. The `/etc/group` file, of course, must be correct for this option to work properly.
- s, -S, -e, -E
Limit the reported information to processes that occur by a specified time. These options can be used with the other options to specify a range of time to which the report of activities will apply. These options require the argument
hr[:min[:sec]]
- s selects processes existing at or after *time*.

- S selects processes starting at or after *time*.
- e selects processes existing at or before *time*.
- E selects processes ending at or before *time*.

PLACE TAB
HERE

SYSTEM
ACTIVITY

BACK OF TAB

Chapter 10

System Activity Package

Contents

1. Introduction	1
2. The system activity counters	1
3. The system activity data collector	2
3.1 The sadc command	2
3.2 The sa1 and sa2 commands	3
4. Setting up the system activity functions	3
5. The system activity report commands	4
5.1 The sar command	5
5.2 The sar command options	7
5.2.1 The -u option	7
5.2.2 The -b option	8
5.2.3 The -d option	9
5.2.4 The -w option	11
5.2.5 The -c option	12
5.2.6 The -a option	13
5.2.7 The -q option	14
5.2.8 The -v option	14
5.2.9 The -m option	15
5.3 The sag command	15
5.4 The time command	16

BLANK PAGE

Chapter 10

System Activity Package

1. Introduction

Two packages provided with your A/UX system allow you to keep track of the details of system operation and usage:

- The system accounting package collects information and generates reports on buffer activity, CPU utilization in general, device activity, and so on. For information on the system accounting package, see Chapter 9, "System Accounting Package."
- The system activity package permits you to keep track of all low-level activity in your system. This chapter discusses the system activity package.

The system activity package provides features for collecting data about the low-level functioning of your system. You can use commands to get information on low-level system activity and to generate reports that summarize this activity. This is accomplished by using various counters to monitor kernel activity. The counters are sampled regularly, and the data is saved in binary format. This data is then used to generate ASCII reports. The information can help you to determine if and where the system may need fine-tuning. You can view the output from these commands immediately or redirect it to a file for future use.

The system activity commands rely on a series of activity counters to gather and sample data and generate a report on system activity. These counters will be described in conjunction with the commands that use them to generate reports.

2. The system activity counters

A series of system activity counters must be working to determine what processes are being run at any given time. These counters, which are located in the operating system kernel, record various activities at

selected times. For example, you can set them to record all processes used at 8:00 A.M. Monday and store the information in a file for review later.

There are several types of counters. Each counter generates various pieces of information, but the same counter may be used by different commands to provide different information. The CPU counter, for instance, reports the prime- and nonprime-time usage in minutes in the accounting report (see Chapter 9, "System Accounting Package"), whereas in the activity report the same counter reports the state(s) the CPU is in: idle, user, kernel, or wait.

In this chapter, each type of counter is explained as it first occurs in relation to the system command that invokes it. Most are explained in relation to the `sar` command, because this is the most useful system activity command. Particular emphasis is placed on those counters that you can use to troubleshoot or fine-tune your system.

If you would like more detailed information about the counters, keep the following in mind:

- The data structure for most counters is defined in the `sysinfo` structure in `/usr/include/sys/sysinfo.h`.
- The system table overflow counters are kept in the `_syserr` structure.
- The device activity counters come from the device status tables.

3. The system activity data collector

The system activity data collector (`sadc`) is the automated data collection feature supplied with your system. Two shell scripts, `sa1` and `sa2`, assemble the data for the reporting functions.

3.1 The `sadc` command

The `sadc` command is used to collect system activity information at regular intervals. It is an executable program that reads the system counters located in `/dev/kmem` and records them in binary format in a file for later sampling by the system activity reporting functions.

The command `sadc` can be used alone or with arguments. It has the format

```
/usr/lib/sa/sadc [t n] [file]
```

When used without arguments, `sadc` sets the startup time. It creates a special record that tells the system to reset the system counters to zero. The same thing occurs when the system is rebooted. When the arguments *t* and *n* are used, `sadc` samples the counters *n* times every *t* seconds and writes the data in binary format to the named *file* or, by default, to the standard output, `/usr/adm/sa/sadd`, where *dd* stands for a given day.

3.2 The `sa1` and `sa2` commands

The `sa1` and `sa2` commands supply the default parameters for the operation of `sadc` and `sar`.

The `sa1` command invokes `sadc` to enter the information gathered by the system counters at the intervals specified in the `/usr/lib/cron/crontab` file into the daily data file `/usr/adm/sa/sadd`. This information is in binary format.

The `sa2` command is a variation of the `sar` command. It reads the data file created by `sa1` and uses it to generate a report in ASCII format. This report is stored in the `/usr/adm/sa` directory in the files named `sardd`. Again, *dd* stands for the day of the report. You can use the `sar` options with the `sa2` command. For a complete explanation of these options, see "The `sar` Command Options," later in this chapter.

4. Setting up the system activity functions

It is a good idea to monitor and record system activity routinely in a standard way for historical analysis. Each of the previous commands initiates activity observations. For these observations to occur, you must first initiate the data collection functions. By automating these functions, you can generate regular system activity reports.

Your system can automatically sample the activity counters and store the information in a file for later reporting. To do this, edit the following lines in `/usr/spool/cron/crontabs/adm` so that they are "commented out" (which is done by placing a number sign before each line):

```
0 * * * 0,6 /usr/lib/sa/sa1
0 18-7 * * 1-5 /usr/lib/sa/sa1
0 8-17 * * 1-5 /usr/lib/sa/sa1 1200 3
```

Next, run the `crontab` command on this newly modified file. This produces records every 20 minutes during working hours and hourly otherwise.

You can generate hourly records during working hours by substituting

```
5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -13600 -A
```

for

```
0 18-7 * * 1-5 /usr/lib/sa/sa1
```

These entries will cause the collection functions to operate. They will not, however, generate reports automatically. It is still the responsibility of the system administrator to check the desired reporting function regularly and to generate the report for current and later review.

5. The system activity report commands

The three commands `sar`, `sag`, and `timex` generate system activity reports.

You can use these commands to observe system activity during

- normal operations
- a controlled stand-alone test of a large system
- an uncontrolled run of a program to observe the operating environment

The `sar` command generates system activity reports in real time and saves the output in a file.

The `sag` command displays system activity in graph form.

The `timex` command is a modified `time` command that reports how long a given command takes to execute and how much user and system time was spent in execution of the command.

These commands and their options and applications are discussed in detail in the next three sections.

5.1 The `sar` command

The system activity reporter command is `sar`. You can use the `sar` command alone or with a series of options. If you enter

```
sar
```

a report on today's CPU activity scrolls across the screen. The output should look like this:

	%usr	%sys	%wio	%idle
00:00:03				
01:00:03	1	1	0	98
02:00:03	1	1	0	98
14:39:12	41	12	4	42
14:59:12	3	5	3	89
15:20:04	14	12	4	70
15:40:04	11	8	3	78
Average	5	5	1	89

If you type

```
sar > sar.output
```

the output from `sar` is stored in the file `sar.output` and doesn't scroll across your screen.

By typing `sar` with the options discussed in the next section, you can sample various counters to view activity at specific times and intervals. By redirecting the output to a file, you can save the information for later review.

The full `sar` command syntax is written in one of these two ways:

```
sar [-u] [-b] [-d] [-w] [-c] [-a] [-q] [-v] [-m] [-A] [-ofile] t [n]
```

```
sar [-u] [-b] [-d] [-w] [-c] [-a] [-q] [-v] [-m] [-A] [-stime] [-etime]  
[-isec] [-efile]
```

The first word is the command itself; the letters within the brackets are flag options that sample different counters. The remaining options allow you to sample certain counters at specified times and save the result for later viewing.

In the command syntax

```
sar [-u] [-b] [-d] [-w] [-c] [-a] [-q] [-v] [-m] [-A] [-ofile] t [n]
```

`sar` extracts the data from a previously recorded file (which you specify with the `-f file` option) or, by default, the standard system activity daily data file `/usr/adm/sa/sadd`, where `dd` stands for a given day. This file appears unreadable when you view it because it is stored in the internal format used by the system reporting functions to prepare reports. The reports themselves are in a readable format.

You can specify the starting and ending times of the report by invoking the `-s time` and `-e time` options, using the format `hh[:mm[:ss]]`.

The `-i` option selects records at *sec* second intervals. Otherwise, all intervals found in the data file are reported. For example,

```
sar -s8:00 -e18:01 -i3600
```

will sample activity at intervals of 3600 seconds, that is, every hour starting at 8:00 A.M. and ending at 6:01 P.M.

In the command syntax,

```
sar [-u] [-b] [-d] [-w] [-c] [-a] [-q] [-v] [-m] [-A] [-stime] [-etime]  
[-isec] [-f file]
```

`sar` invokes the data collection program `sadc` to sample the system activity counters every *t* seconds for *n* intervals and generates a system activity report.

If you specify the `-o` option, `sar` saves the output in binary format into *file*.

If you supply no frequency arguments, `sar` generates system activity reports for the time interval specified in the existing data file. On this system, that time period is from 8:00 to 18:00, and the data is sampled every hour (see "Setting Up the System Activity Functions," earlier in this chapter). Unless you redirect the output to a file, it scrolls across the screen.

Note: All reports you generate with the options listed print a time stamp for each entry. This time stamp appears in the first column of each report section in the format `hh:mm:ss`.

When used without options, the `sar` command generates a report about CPU activity only. This is the same report you receive if you enter

```
sar -u
```

The system activity package automatically generates a report containing all of the options listed in the following sections. The report is stored in the `/usr/adm/sa` directory, in the files listed as `sardd`. These files are the binary representations of the `sar` reports. You can get the same report by typing

```
sar -A
```

This generates the same information as

```
sar -udqbwcaym
```

but takes fewer keystrokes.

5.2 The `sar` command options

The options you can use with the `sar` command are listed in the following subsections.

5.2.1 The `-u` option

Use the `-u` option to get a report on CPU utilization. This is the default option, the option the system selects if none is specified. If you enter

```
sar -u
```

a report similar to the following one is displayed:

07:00:02	%usr	%sys	%wio	%idle
08:00:04	0	1	0	99
09:00:02	10	7	3	80
11:00:07	8	8	4	81
12:00:05	15	10	4	71
13:00:03	12	10	2	76
15:00:08	31	16	4	48
16:00:06	39	15	4	43
18:00:02	1	1	0	97
19:00:02	0	1	0	99
Average	12	8	2	78

The column headings display the following:

%usr amount of time running in user mode
 %sys amount of time running in system mode
 %wio idle time with process waiting for block I/O
 %idle idle time
 Average daily averages

All of the information under these headings is sampled each hour to produce the report. The headings correspond to the four CPU counters: user, kernel, wait for I/O completion, and idle. These counters are increased by one (incremented) each time the clock calls for an interrupt (a count on the system), which occurs 60 times per second.

5.2.2 The -b option

The -b option reports buffer activity. This option works by accessing three sets of read and write counters.

- lread and lwrite (logical read, logical write)
- bread and bwrite (block read, block write)
- phread and phwrite (physical read, physical write)

If you enter

```
sar -b
```

the system generates a report organized in columns as follows:

```
bread/s, brwrit/s
```

Samples the bread and brwrite counters, giving the transfers

of data per second between the system buffers and the disk.

lread/s, lwrit/s

Samples the lread and lwrite counters, giving the number of times the system buffers are accessed.

phread/s, phwrit/s

Samples the phread and phwrite counters, giving the number of data transfers via raw devices.

%rcache, %wcache

Gives the ratio of buffer reads (bread) to logical reads (lread) and buffer writes (bwrit) to logical writes (lwrit) in what is called the cache hit ratio. The cache hit ratio reports if files are being accessed from the buffers or if the information has to be retrieved from the disk itself. A low ratio might suggest that more efficient buffering could increase system response time in a certain area.

5.2.3 The -d option

The -d option reports device activity. This option is unreliable, and the information it provides is usually inaccurate. When you select this option, you can view information on block devices, such as the disk or tape drives.

If you enter

```
sar -d
```

the system generates a report organized in columns as follows:

Device

The device being sampled.

%busy

Samples the device counters and displays the time, expressed as a percentage of total report time, during which the device was engaged in transferring data.

avque

Displays the average number of requests waiting to be processed.

r+w/s

Displays the number of data transfers to or from the disk or tape drive.

blks/s

Displays the number of bytes transferred and counted in block-sized units.

await

Displays the number of milliseconds that transfer requests have to wait in the queue before being processed.

avserv

Displays the average time it takes to service the request.

The figures in these columns represent a sampling of a combination of I/O activity counters and character counters. The `%busy` column, for example, represents a sampling of the `io_ops` counters and the `io_act` counters. The `blks/s` column represents a sampling of the `io_bcnt` counters. In addition, the `avserv` and `await` columns represent a sampling of the `io_act` and `io_resp` counters. These counters are explained in greater detail later in this section, and some suggestions for fine-tuning are given.

Each disk or tape device has four counters to record activity. The activity information is kept in the device status table. Whenever an I/O request occurs, `io_ops` (I/O operations) is incremented. It keeps track of block I/O, swap I/O, and physical I/O.

Transfers between the device (particular disk or tape drive) and memory are recorded in 512-byte blocks by `io_bcnt` (I/O block counters). The `io_act` and `io_resp` are particularly useful I/O counters. By time ticks, they measure the time it takes a device to receive, process, and transmit a request, summed over all I/O requests for the device. The `io_act` counter measures the active time, which is the time during which the device is actively engaged in seeking, rotating, and transferring data (all measured by different counters and combined into one active count).

The `io_resp` counter measures the total elapsed time between the time the request is received in the queue and the time it is completed. By looking at the ratio between active time and response time, you can

determine if the disk and tape devices are being put to their best use. For example, if one device is so heavily used that response time is significantly increased, perhaps you can shorten system reaction time by transferring some frequently used information to another, less used device. Also, if the active time counter for a device is high compared with the number of requests on the system, perhaps you could load the file systems on the device differently for more effective access.

5.2.4 The -w option

The -w option reports swapping and switching activity. This option uses the `swpin` and `swpot` counters. (The column headings that appear in the report, `swpin` and `swpot`, are abbreviations.) These counters are incremented each time the system receives a request initiating a transfer to or from the swap device. The swap device is a disk partition used as a "holding area" for processes that are not currently running. When main memory is full, processes that are not currently running are swapped out (transferred) to the swap device. When the CPU is ready to work on a process that is in the swap device, the process is swapped back into memory.

The amount of data swapped in and out is measured in blocks and counted by the `bswapin` and `bswapout` counters. The data collected by these counters is displayed in the column headings `bswin` and `bswot`.

The figures for `swpin` are usually higher than those for `swpot`. This peculiar asymmetry arises from programs with the "sticky bit" set, which keeps a program on a contiguous area of the swap device. Therefore, moving the program back and forth between memory and the swap device is more efficient if the sticky bit were not set.

If you enter

```
sar -w
```

the system generates a report organized in columns as follows:

`swpin/s`

Reports the number of transfers from the swap area on disk to main memory.

`swpot/s`

Reports the number of transfers from memory to the swap area.

bswin/s, bswot/s

Reports the number of bytes transferred.

pswch/s

Reports the number of process switches that have occurred.

5.2.5 The -c option

The **-c** option reports system calls. This option accesses the **pswitch** and **syscall** counters. These counters are related to the management of multiprogramming. This occurs when one process, the parent process, calls (forks and executes) another program, the child process. While the parent waits, the child performs its task, terminates, and reinvokes the parent process, which continues.

The **syscall** counter is incremented every time a system call occurs. Certain system calls — the **read**, **write**, **fork**, and **exec** calls — are counted individually in **sysread**, **syswrite**, **sysfork**, and **sysexec**.

The **pswitch** counter keeps tracks of the number of times the switcher was invoked. The switcher is invoked when the program running could not complete its intended process.

The following situations cause **pswitch** to be incremented:

- a system call that resulted in a roadblock
- an interrupt that caused the awakening of a higher-priority process
- a 1-second clock interrupt

To check the number of system calls, type

```
sar -c
```

The first five columns of the resulting output give information on the number of system calls made. The first column (**syscall/s**) gives the total number of system calls; the next four columns give the number of specific system calls for **read** (**sread/s**), **write** (**swrite/s**), **fork** (**fork/s**), and **execute** (**exec/s**) system calls, respectively. The last two columns give the number of characters transferred by the **read** (**rchar/s**) and **write** (**wchar/s**) system calls, respectively.

5.2.6 The -a option

The -a option reports on use of file access system routines. Avoid using this option because this information is usually inaccurate. This option of `sar` checks the following:

- the number of times a file's inode number is requested
- the number of file path searches
- the number of directory blocks read by the system

If you enter

```
sar -a
```

the system samples the file access counters and generates a report showing the number of times each of the file access routines was performed. The report is organized into columns as follows:

`iget/s`

Measures the number of requests for the inode number that corresponds to a particular file. The `iget` routine is used to locate the inode entry (i-number) of a file. See Chapter 8, "Checking the A/UX File System: `fsck`," for an explanation of i-numbers and inodes. The `iget` routine first searches the in-core (main memory) inode table. If the inode entry is not in the table, `iget` gets the inode from the file system where the file resides and makes an entry in the inode table.

`namei/s`

Measures the number of requests for a file system path search. The `namei` routine performs file system path searches. It searches the various directory files to get the associated i-number of a file corresponding to a special path. Like other file access routines, `namei` calls `iget` to find the i-number of the file it is searching for. Therefore, counter `iget` is always greater than counter `namei`.

`dirblk/s`

Measures and records the number of directory block read requests issued by the system. Dividing the directory blocks read by the number of `namei` calls results in an estimate of the average path length of files. A long path length may indicate a

significant number of subdirectories. Rearranging the file structure to move the more commonly accessed files higher up the path would correct this problem.

Each time one of these routines is called, the respective counter is incremented.

5.2.7 The -q option

The -q reports on queue activity. At every one-second interval, the clock routine examines the process table to see whether any processes are queued and ready. If so, the counter `runocc` is incremented and the number of processes waiting is added to the `runque` counter. While this is happening, the clock routine also checks the process status of the swapper. If the swap queue is occupied, the counter `swapocc` (swap occupied) is incremented and the number of processes waiting in the queue is added to the `swapque` counter.

If you enter

```
sar -q
```

the system reports on the average time a process is queued before it is acted on. The report lists the average queue length while the queue is occupied (the columns ending in -sz) and the percentage of time the queue is occupied (the columns beginning with %). It is broken down into two main parts: the run queue (the `runq` columns), which lists the processes in memory and runnable; and the swap queue (the `swp` columns), which lists the processes swapped out but ready to run.

5.2.8 The -v option

The -v option reports the status of text, process, inode, and file tables. The information provided is usually inaccurate. When an overflow occurs in any of the inode, file, text, or process tables, the corresponding counter (`inodeovf`, `fileovf`, `textovf`, or `procovf`) is incremented. These indicate resource problems with tables or the size of memory.

You can use the -v option of the `sar` command to discover the size of tables and any table overflows.

If you enter

```
sar -v
```

the system produces a report in which the first five columns show the number of used and available entries in each table. This information is typically given for one-hour intervals. The measurement is taken once at the sampling point. The last four columns give the number of overflows that occur between the sampling points.

5.2.9 The `-m` option

The `-m` option reports message and semaphore activities. This option of the `sar` command reports which processes requested the operating system to send information directly to another process.

If you enter

```
sar -m
```

the system generates a report on message and semaphore activity. The message and semaphore columns (`msg/s` and `sema/s`) reflect the most basic input/output operations of the system. These "primitives" (for example, `read` and `write`) are called by other programs, which use them as building blocks to complete their processes. The message primitives keep track of interprocess communications; that is, the number of times one process asks the operating system to send information directly to another process. The semaphore primitives synchronize the actions of various processes and facilitate the use of shared resources.

5.3 The `sag` command

The system activity graph command is `sag`. It displays the system activity data that was created by a previous run of the `sar` command and stored in binary format.

You can plot the graph using any single column or combination of columns. It can prepare cross-plots or time plots.

A graphics package that can invoke the `graphics` and `tplot` commands must reside on the system for you to print a system activity graph. Unfortunately, very few terminals are supported. The Macintosh II and common emulators such as the VT100 are not among those supported. See `sag(1G)` in *A/UX Command Reference*.

5.4 The `timex` command

The `timex` command is an extension of the `time` command; see `time(1)` and `timex(1)` in *A/UX Command Reference*. It times a command and reports process data and system activity. The command you are tracking is executed, and the elapsed time, user time, and system time spent in execution are reported in seconds.

The options available with `sar` are also available with `timex`. However, `timex` tracks one command, whereas `sar` tracks all commands. The output of the `timex` command is easier to understand than the output of `sar`, and it is also generally more reliable.

Normally, you use the `timex` command to measure a single command. If you want to measure multiple commands, combine the commands in an executable file and time the file. You can also do this by entering

```
timex sh -c "cmd1; cmd2;...;"
```

This allows `timex` to measure the user and system times consumed by all the commands as if they were one single command. See `sh(1)` in *A/UX Command Reference*.

Because process records associated with a command are selected from the accounting file `/usr/adm/pacct`, background processes that have the same user ID, terminal ID, and execution time window are included in the totals given.

You can specify options to list or summarize process accounting data for the command and its children and to report the total system activity during the execution interval. If you don't specify any options, `timex` behaves exactly as the `time` command does.

The syntax for the `timex` command is

```
timex [-pos] command
```

The `timex` command options are as follows:

- p Lists the process accounting records for the specified command. This option has six suboptions:
 - f Prints the `fork/exec` flag and system exit status.
 - h Reports the fraction of total available CPU time the process consumes during its execution and suppresses reporting of the mean memory size.

- k Reports the total kcore-minutes and suppresses reporting of memory size.
- m Reports the mean core size.
- r Reports the fractional representation of CPU factor (user time) over system time plus user time.
- t Reports separate system and user CPU times.
- o Reports the total number of blocks read or written and total characters transferred by the command selected and all its children.
- s Reports the total system activity during the execution interval of the command, not just the activity resulting from the command specified. All the data items listed in `sar` are reported.

BLANK PAGE

PLACE TAB
HERE

UUCP

BACK OF TAB

Appendix A

The UUCP System

Contents

1. Introduction	1
2. The components of UUCP	2
2.1 Directories	3
2.2 Executable files	4
2.3 Script files	5
2.4 A/UX system files	6
2.5 UUCP system files	7
2.6 Statistical files	10
2.7 Sequence files	11
2.8 Log files	11
2.9 Audit files	12
2.10 Spool files	12
2.11 Lock files	12
2.12 Status files	13
2.13 Temporary files	13
2.14 Backup files	14
3. Setting up the L-devices and L.sys files	14
3.1 The /usr/lib/uucp/L-devices file	14
3.2 The /usr/lib/uucp/L.sys file	15
4. Interactive file transfer	18
5. Automatic file transfer	20
5.1 Preparing the systems	20
5.1.1 The system node name	20
5.1.2 Dial-in and dialout ports	20
5.1.3 Generic uucp logins	21
5.1.4 System-specific uucp logins	21
5.2 Receiving mail and files	22
5.3 Sending mail or files	24

5.4	Cleaning up	24
6.	Security and other tips	26
6.1	Controlling logins and file system access	26
6.2	Conversation count checking	27
6.3	Controlling file forwarding	28
6.4	Controlling remote command execution	28
6.5	File permissions	28
6.6	The nuucp login environment	30
6.7	Suggested links	32
6.8	Troubleshooting uucp	32

Tables

Table A-1.	Expect-send strings for the Apple Personal Modem	18
------------	---	----

Appendix A

The UUCP System

1. Introduction

One of the most important features of A/UX is its ability to transfer information among A/UX systems (and among other systems derived from UNIX). The standard A/UX communication package, called UUCP (for "UNIX to UNIX copy"), is the mechanism generally used for this function. The UUCP package permits mail, command execution, and file transfers among A/UX computers. These functions are useful whenever related pieces of information and/or users are located on separate computers. For example, mail can be used to communicate with other companies, file transfer can be used to copy programs directly from one computer to another, and remote command execution can be used to print documents on a remote computer that has an attached laser printer.

When two or more computers can communicate, the computers and the communication channels (for instance, telephone lines) between them are considered parts of a computer network. Each computer is a node in the network; the communication channels are network links. Communication between nodes can take place using one of two primary methods: interactive or automated. **Interactive communication** requires a user to log in to the remote system and issue commands to initiate command execution and file transfer. **Automated communication** requires some method by which the local computer can communicate with the remote computer(s) without user intervention.

This appendix describes the many commands, scripts, and spooling directories involved in UUCP and then introduces a step-by-step procedure for setting up UUCP to handle interactive logins and file transfer between UNIX nodes. Next, the procedure for setting up mail to a remote UNIX node is described. The appendix concludes with hints for modifying, expanding, and tightening the security of UUCP on your system.

2. The components of UUCP

This section discusses the user files and directories used in the running of UUCP and the system files and directories used in the administration of UUCP.

The files can be grouped into the following categories:

- **binary:** binary executable files used by UUCP
- **script:** shell scripts used as commands and for system maintenance
- **A/UX system:** A/UX system files also used by UUCP
- **uucp system:** files used by UUCP for system configuration
- **statistical:** files used by UUCP for storage of statistical information
- **sequence:** files used for storing sequential number information
- **log:** files used for logging information on requests and connections
- **audit:** files used for storing debugging information
- **spool:** files used to spool requests
- **lock:** files used to lock UUCP system files and prevent simultaneous updating or accessing
- **status:** files used to store status of connections to specific systems
- **temporary:** files used for temporary storage of information
- **backup:** files used for keeping backups of recent log files

After a discussion of the directories involved, this section focuses on each of these categories and the files that constitute them.

2.1 Directories

The following directories are used exclusively by uucp:

/usr/lib/uucp

Used for storage of UUCP system files that are not temporary in nature. This includes script files run by `cron` and executable binary files that are forked by UUCP processes.

/usr/spool/uucp

Used for storage of UUCP files that are temporary in nature. This includes log files and spool files.

/usr/spool/uucp/.XQTDIR

Used by `uuxqt` for temporary storage of files used in remote command execution by `uux`.

/usr/spool/uucppublic

The directory reserved for public usage when files are sent from one system to another. The directory grants read, write, and execute permission to every user of the system (and for this reason is said to be for public usage). All files that are to be sent to the computer should be sent to this directory. Some subdirectories are used for specific purposes.

/usr/spool/uucppublic/user

Where *user* is a specific user name, used by the system to place files that could not otherwise be transferred because of permission problems.

/usr/spool/uucppublic/receive

Along with all of its subdirectories, used exclusively by the programs `uuto` and `uupick`.

/usr/spool/uucppublic/receive/user

Where *user* is a known user on the system, used to build directories for files from other systems sent by `uuto`.

/usr/spool/uucppublic/receive/user/system

Where *user* is a known user on the computer and *system* is a remote computer, used to store files sent from *system* to *user* with `uuto`. This is the directory where `uupick` will find files.

2.2 Executable files

The following files are the binary executable files used by UUCP for common usage and administration:

/usr/bin/uucp

The user command that copies files from system to system. Forwarding may be allowed through intermediate nodes. See *uucp(1C)* in *A/UX Command Reference*.

/usr/bin/uulog

May be used as a user command or a system maintenance command. As a user command, it presents logged information about *uucp* or *uux* requests by either system name or user name. As a maintenance command, it appends any temporary log files to the permanent log file. See *uucp(1C)* in *A/UX Command Reference*.

/usr/bin/uuname

A user command whose output is either the local node name or a list of the node names of all computers with which the system can communicate. See *uucp(1C)* in *A/UX Command Reference*.

/usr/bin/uustat

May be used to display the status of *uucp* jobs or connections and can be used to cancel certain jobs. See *uustat(1C)* in *A/UX Command Reference*.

/usr/bin/uusub

May be used as a user command or a system maintenance command. As a user command, it displays the statistics on *uucp* connections or traffic. As a maintenance command, it flushes the statistics files, adds new systems for statistics gathering, or calls other systems. See *uusub(1M)* in *A/UX System Administrator's Reference*.

/usr/bin/uux

A user command that executes commands on a remote system and takes care of transferring all files necessary for the command. See *uux(1C)* in *A/UX Command Reference*.

/usr/lib/uucp/uucico

Forked by *uucp* or *uux* to call other systems and do all the

work. It may be called directly from shell scripts started by `cron` to scan for work needing to be done. It is also executed directly when a remote system logs in.

/usr/lib/uucp/uuclean

A system maintenance command usually called from shell scripts started by `cron` to remove old files from uucp directories. See `uuclean(1M)` in *A/UX System Administrator's Reference*.

/usr/lib/uucp/uuxqt

Forked by `uux` or `uucico` to execute the scripts that `uux` sets up to perform remote command execution.

2.3 Script files

The following are script files used as normal commands and files used for system maintenance:

/usr/bin/uupick

A user shell script used to pick up files that have been sent to a user from another system via `uuto`. The appropriate public subdirectories are searched, and the user may accept or reject files for copying to another directory. See `uuto(1C)` in *A/UX Command Reference*.

/usr/bin/uuto

A user shell script that sends files or entire directories via uucp to a user on another system. In the case of directories, entire subtrees are sent by calling uucp repeatedly. See `uuto(1C)` in *A/UX Command Reference*.

/usr/lib/uucp/uushell

A login shell script used to set the time zone environment variable before executing `uucico`. This will cause log file entries to show the correct time for remotely initiated requests and connections. It should be the login shell for all uucp connections.

/usr/lib/uucp/uudemon.day

A maintenance shell script started by `cron` every night to perform UUCP maintenance. It is normally used to remove old files from UUCP directories and trim status and statistics files,

that is, to eliminate old entries and thus prevent the files from growing too large.

/usr/lib/uucp/uudemon.hr

A maintenance shell script started by `cron` every hour to perform UUCP maintenance. It is normally used to append temporary log files to permanent log files and to check for spooled work.

/usr/lib/uucp/uudemon.wk

A maintenance shell script started by `cron` every week to perform uucp maintenance. It is normally used to store week-old log files and remove two-week-old log files.

The descriptions of the `uudemon` scripts are intended to demonstrate their typical uses. You decide when each of these scripts is actually run via its respective `crontab` entry. You can also easily change what these scripts actually do by editing the script files.

2.4 A/UX system files

The following system files are used by UUCP:

/etc/group

Keeps track of group IDs for user names. UUCP uses these user names to allow remote systems to log in. See Chapter 3, "User Administration."

/etc/inittab

Generates `gettys` for ports. These `gettys` must be enabled for dial-in ports and disabled for dialout ports. See "Dial-in and Dialout Ports," later in this appendix.

/etc/passwd

Keeps track of user names and passwords. UUCP uses these to allow remote systems to log in and transfer files. Both the home directory and the login shell are specified for each user name. See Chapter 3, "User Administration."

/usr/spool/cron/crontabs/uucp

Tells the `cron` process when to schedule the execution of programs. UUCP uses this to schedule hourly, daily, and weekly shell scripts to perform maintenance. See Chapter 9, "System Accounting Package."

2.5 UUCP system files

UUCP uses the following system files for configuration:

/usr/lib/uucp/ADMIN

Stores additional information about each system that appears in the `L.sys` file. You can display this description along with the system names by using the command `uuname -v`.

/usr/lib/uucp/FWDFILE

Stores a list of system names to which files may be forwarded. These are a subset of the `L.sys` system names and do not restrict the final destination, just the next system in the path. Each line in this file takes the form

system[, user, user]

where *system* is the name of a system to which a communication can be forwarded, and the optional list of *users* separated by commas represents the login names of users in the system to which the communication can be forwarded.

/usr/lib/uucp/L-devices

Stores the names of the ports that are connected to modems or directly to other systems. The file contains the attributes, such as baud rate, of each of these ports. Each line in this file takes the form

type line device speed [protocol]

where

- type* can be either `DIR`, indicating that the line is directly connected to another system (this includes modem connections), or `ACU`, indicating that the line uses an automatic calling unit
- line* is the device name of the line (for instance `ttty0` if the modem is connected to port `/dev/tty0`)
- device* is the device name of the ACU if one is specified in the *type* field (otherwise, a placeholder `[0]` must be used in this field)

speed is the line speed for the connection, as measured in baud

protocol is an optional field that needs to be filled only if the connection is for a protocol other than the default protocol

Here is a typical entry in the L-devices file:

```
DIR tty0 0 1200
```

`/usr/lib/uucp/L-dialcodes`

Stores dial-code abbreviations for L.sys. Each abbreviation is associated with a dial sequence of numbers (typically an area code). Each line in this file takes the form

abbreviation dial-ing-sequence

where *abbreviation* stands for an arbitrary abbreviation of an area code or telephone exchange number, and *dial-ing-sequence* stands for the telephone number that should be dialed after the abbreviation.

Here is a typical entry in this file:

```
sfba 415
```

The abbreviation *sfba* stands for San Francisco Bay Area, and 415 is the area code.

`/usr/lib/uucp/L.cmds`

Stores a list of command names that uux is permitted to execute. If a command name (including *rmail*) is not listed in this file, it cannot be used for remote execution. Each line in this file takes the form

cmd

where *cmd* is the name of any command you want uux to be able to execute in your system.

`/usr/lib/uucp/L.sys`

Stores information on connecting to remote systems. Each line represents a way to connect to another system. If more than one

line exists for a particular system, each line is tried sequentially until a connection is made. Each line in this file takes the form

system time device class phone login

where

system is the name of the remote system;

time is the time(s) at which that system can be called;

device is either the name of the port (if it is a DIR connection) or the keyword ACU;

class is the speed of the connection in baud;

phone is the phone number to be called, expressed either in an all-numeric sequence or as an alphabetic abbreviation, and specified in a corresponding line in the L-dialcodes file;

login is an *expect-send-expect* sequence as specified in "Automatic File Transfer," later in this appendix.

Here is a typical entry in this file:

```
doosy Any tty0 1200 tty0 "" ATDTsfba5551212~M
ogin:~@-ogin:-EOT-ogin:-BREAK-ogin:U_mickey ssword:mouse
```

/usr/lib/uucp/ORIGFILE

Stores a list of system names and users from which files may be forwarded. Each entry refers to the system that was the originator of the request and not the most recent system in the path. Each line in this file takes the form

system[, user, user]

where *system* is the name of a system whose communication the system is willing to forward, and the optional list of *users* separated by commas represents the login names of users in that system whose communication can be forwarded. Note that *system* represents the name of the system that originated the communication, not the name of the last system that forwarded the communication.

/usr/lib/uucp/USERFILE

Stores access permissions. These include pathname prefixes that are accessible by local users and remote systems. Remote system login names must appear here with an optional call-back facility. Each line in this file takes the form

[login], system [c] path [path]

where *login* is the login name of a user or a remote computer; *system* is the system name of a remote computer; *c* is an optional flag that, as a security measure, requires that the remote computer be called back before any further communication takes place; and *path* is a pathname prefix constraining file access to only those files preceded by the prefix. If the *login* field is empty (a null login), any user can access the specified path.

For an example, the lines

```
root, /  
, /usr/spool/uucppublic
```

permit any user at any remote computer to transfer any files from /usr/spool/uucppublic, but only a user with the login name of root can transfer any file, because all filenames ultimately begin with /.

2.6 Statistical files

UUCP uses the following files to store status and statistical information:

/usr/lib/uucp/L_stat

Stores the latest connection status of each remote system. You can display this information using the `uustat` command.

/usr/lib/uucp/L_sub

Stores connection statistics for each remote system. You can display this information using the `uusub` command.

/usr/lib/uucp/R_stat

Stores the status of each uucp request. You can display this information using the `uustat` command.

`/usr/lib/uucp/R_sub`

Stores traffic statistics for each remote system. You can display this information using the `uusub` command.

2.7 Sequence files

UUCP uses the following files for storing sequence information:

`/usr/lib/uucp/SEQF`

Stores the sequence number, which is incremented by one for each uucp request. This is a four-digit number used in generating the names of the spooled files.

`/usr/lib/uucp/SQFILE`

Stores a conversation count for remote systems. These systems will be a subset of the systems in `L.sys`. The remote system must also have an entry for your system in its `SQFILE`. If a connection is made but the counts in the two files do not agree, the login attempt fails.

`/usr/lib/uucp/SQTMP`

Temporarily stores the `SQFILE` used by `uucico`.

2.8 Log files

UUCP uses the following files for logging information on UUCP requests and connections:

`/usr/spool/uucp/ERRLOG`

Logs uucp errors generated when `uucico` fails on a file transfer. If the `-x` option is used with `uucico`, the errors do not appear in this file.

`/usr/spool/uucp/LOGDEL`

Used by `uuclean` to log files that have been removed.

`/usr/spool/uucp/LOGFILE`

Logs the status of calls and uucp requests. During execution of uucp requests, log information is normally appended to the file. If more than one uucp process is active at a time, the information is logged to temporary files. You can use the `uulog` command to display portions of this file as well as append temporary versions to it.

`/usr/spool/uucp/SYSLOG`

Logs the number of bytes sent and received during each connection and the duration of the connection in seconds.

2.9 Audit files

UUCP uses the following files to store debugging information:

`/usr/spool/uucp/AUDIT`

Stores debugging information from the remote `uucico` process. It is created every time `uucico` is run as a login shell.

`/usr/spool/uucp/AUDIT.system`

Where *system* is the name of a remote computer, stores debugging information when the remote computer calls with `uucico` and the `-x` option.

2.10 Spool files

UUCP uses the following files to spool requests for work to be done:

`/usr/spool/uucp/C.systemxxddd`

Stores information for `uucico` on which system to connect and which source and destination filename to transfer. It is created whenever `uucp` or `uux` is executed.

`/usr/spool/uucp/D.systemxxddd`

Stores data files for transfer. This file is created when you use the `-c` option with `uucp`.

`/usr/spool/uucp/X.systemxxddd`

Stores information used to execute remote commands. This file is created prior to running `uuxqt`, which reads it.

In these filenames, *system* is the name of the remote computer, *xx* are two ASCII characters representing the priority of the work, and *ddd* is the sequence number from SEQF.

2.11 Lock files

UUCP uses the following as lock files to prevent simultaneous update of UUCP system files:

`/usr/spool/uucp/LCK..system`

Where *system* is the name of a remote computer, prevents more than one simultaneous connection to that computer. Notice the

two dots (..) in the name. These are an integral part of the filename.

`/usr/spool/uucp/LCK..tynn`

Where *ty*nn is the name of a port used for dialout or direct connection, prevents more than one `uucico` process from using the port at the same time. Notice the two dots (..) in the name. These are an integral part of the filename.

`/usr/spool/uucp/LCK.LOG`

Prevents simultaneous update of the log files.

`/usr/spool/uucp/LCK.LSTAT`

Prevents simultaneous update of `L_stat`.

`/usr/spool/uucp/LCK.LSUB`

Prevents simultaneous update of `L_sub`.

`/usr/spool/uucp/LCK.RSTAT`

Prevents simultaneous update of `R_stat`.

`/usr/spool/uucp/LCK.RSUB`

Prevents simultaneous update of `R_sub`.

`/usr/spool/uucp/LCK.SQ`

Prevents simultaneous update of `SQFILE`.

`/usr/spool/uucp/LCK.SEQL`

Prevents simultaneous update of `SEQF`.

`/usr/spool/uucp/LCK.XQT`

Prevents simultaneous remote command execution by `uuxqt` (see "Executable Files," earlier in this appendix).

2.12 Status files

UUCP uses the following file to store the status of connections to specific systems:

`/usr/spool/uucp/STST.system`

Where *system* is the name of a remote computer, used to store the status of a connection that fails.

2.13 Temporary files

UUCP uses the following files for temporary storage:

`/usr/spool/uucp/LTMP.pid`

Used to store LOGFILE entries temporarily when more than one uucp process is executing; *pid* is the process ID. Use the `uulog` command to append these files to LOGFILE.

`/usr/spool/uucp/TM.pid.ddd`

Temporarily stores these data files until the transfer is complete; *pid* is the process ID, and *ddd* is the number of this file in the current transfer. If the transfer is successful, the file is moved to the correct destination; otherwise, it is removed.

2.14 Backup files

UUCP maintenance scripts create the following files to keep backups of recent log files:

`/usr/spool/uucp/Log-WEEK`

Stores the LOGFILE entries for the current week to date. The shell script `uudemond.day` appends the current LOGFILE to `Log-WEEK` every night.

`/usr/spool/uucp/o.Log-WEEK`

Stores the LOGFILE entries for the entire previous week. Between `Log-WEEK` and `o.Log-WEEK`, the system will have a backup of one to two weeks of LOGFILE entries.

`/usr/spool/uucp/o.SYSLOG`

Stores the SYSLOG entries for the entire previous week.

3. Setting up the L-devices and L.sys files

You will become familiar with many files as you work with the UUCP system. Two are of particular importance because they affect both interactive and automated file transfers.

3.1 The `/usr/lib/uucp/L-devices` file

The `L-devices` file contains information about what devices the computer can use to communicate with the outside world (modem, automatic calling unit, and so on). If you already have a modem on your system, the file should contain information about what port it is attached to. If you don't have a modem attached, you will have to attach one to communicate with other computers. See Chapter 7, "Managing Other Peripheral Devices" and "Dial-in and Dialout Ports," later in this appendix, for a review of how to attach a modem.

Each line in the `L-devices` file has the form

type line device speed [protocol]

where

- type* can be either `DIR`, indicating that the line is directly connected to another system (this includes modem connections), or `ACU`, indicating that the line uses an automatic calling unit
- line* is the device name of the line (for instance `tty0` if the modem is connected to port `/dev/tty0`)
- device* is the device name of the ACU if one is specified in the *type* field (otherwise, a placeholder `[0]` must be used in this field)
- speed* is the line speed for the connection, as measured in baud
- protocol* is an optional field that needs to be filled only if the connection is for a protocol other than the default protocol

Consider this entry from an `L-devices` file:

```
DIR tty0 0 1200
```

`tty0` is the name of the port to which the modem is attached. If your modem is attached to another port, you should substitute its name for `tty0`.

1200 is the speed, in baud, at which the modem will communicate. If you have a 300/1200 selectable modem, you should have two lines like these in your `L-devices` file:

```
DIR tty0 0 1200
DIR tty0 0 300
```

3.2 The `/usr/lib/uucp/L.sys` file

The `L.sys` file contains information that your system uses to call other computers.

Note: The `L.sys` file contains information vital to the security of your neighboring UUCP sites. Keep its permissions closed to reading by unauthorized parties.

Here is a typical `L.sys` file entry:

```
doosy Any tty0 200 tty01 "" ATDT5551212^M\  
ogin:~-@-ogin:~EOT-ogin:~BREAK-ogin:U_mickey ssword:mouse
```

In this entry, `doosy` is the name of the system this entry allows you to call. Every system has a name that identifies it.

`Any` indicates that your system can call `doosy` at any time. You will find out how to restrict these times later. Once again, `tty0` is the port to which the modem is attached, and `1200` is the speed at which the modem will communicate. The second `tty0` is a placeholder for a telephone number.

The rest of the line has the form

`expect-send-expect-send`

where *`expect`* is what your computer expects the other computer to transmit, and *`send`* is what your computer will transmit to the remote site. The `""` in the example is a null string that will expect nothing. `ATDT5551212^M` is sent to the modem, telling it to dial 555-1212. (This assumes that you are using a Hayes-compatible modem. For any other type, you would substitute whatever command would cause the modem to dial the number. See the modem owner's manual.)

Note: `^M` is CONTROL-M. It is not enough to press CONTROL-M. To enter `^M` into a file when using the `vi` editor, press CONTROL-V. Then press CONTROL-M. This tells the computer to send a carriage return. This is not the same as typing `^m` (caret-m) or UP ARROW-m.

The second *`expect`* field is then broken up into

`expect-send-expect-send-expect-send-expect`

In other words, the simplest form of *`expect-send-expect-send`* for the rest of the line would be `ogin: U_mickey ssword: mouse`. This tells `uucp` to expect a prompt ending with `ogin:~`. When it gets the prompt, it sends `U_mickey`. Then it expects `ssword` and replies with `mouse`. The strings `ogin:~` and `ssword:~` are used because most computers send a string ending in either `Login:~` or `login:~`.

each port. After you enter a login, most computers respond with either Password or password. Unfortunately, things are not so simple. For instance, this confusing expect string

```
ogin:~@-ogin:-EOT-ogin:-BREAK-ogin:
```

is of this form

```
expect-send-expect-send-expect-send-expect
```

It means this: expect "ogin:~"; if that does not happen, wait (that's what the @ means) and expect "ogin:". If that does not happen, send EOT, and so on.

Why the login Umickey? The U_ is an ad hoc convention for uucp logins, but you need not adhere to it. This login name can be any name on which you and the remote system administrator agree, and it is often the name of the remote system.

While A/UX supports dial-ing on assorted modems via the /etc/remote and /etc/phones files, (see remote(4) and phones(4) in *A/UX Programmer's Reference*), you may wish to dial an unsupported modem.

The following L.sys entry shows how to accomplish this, using the Apple Personal Modem (APM) as an example:

```
foo Any tty0 1200 tty0 "" atdt1234567\r 1200 \r
    ogin:-BREAK-ogin: Umine ssword: passwd4foo
```

Note: Although these lines have been wrapped onto two lines here, they must appear on a single line in the A/UX L.sys file.

This entry means that machine foo may be called using tty0 at 1200 bits per second, at any time. UUCP is told to expect and send the following strings:

Table A-1. Expect-send strings for the Apple Personal Modem

<i>expect</i>	<i>send</i>	Comments
""		Don't wait for anything.
	atdt1234567\r	Send APM command line.
1200		Wait for "CONNECT 1200".
	\r	Send a carriage return.
ogin:-BREAK-ogin:		Wait for ... login.
		Send a break, if necessary.
	Ubar	foo knows us as Ubar.
ssword:		Wait for password:.
	passwd4foo	Send the password...

4. Interactive file transfer

At this point you can already start transferring files to and from another system, provided that the other system is set up so that you can log into it.

To dial out, you must make sure that the modem port does not have a `getty` running on it, that is, that it is not working as an incoming modem. To do so, you must establish your modem either as an outgoing modem only or as both a dialout and a dial-in modem. See Chapter 7, "Managing Other Peripheral Devices." To call up the other computer, you can use the `cu` command (see Chapter 5, "Using `cu`," in *A/UX Communications User's Guide*). You can, for instance, enter.

```
cu -lline dir
```

where *line* is the name of the port to which the modem is attached and *dir* ensures that `cu` uses that line. In this case, `cu` looks in the `L-devices` file for the appropriate speed at which to communicate. Once `cu` finds that speed, it informs you by printing the message

```
Connected
```

on the screen. This means that you are connected to the modem, and you can now instruct the modem to dial out, for instance,

```
ATDT5551212
```

if your modem is Hayes-compatible. If the computer at the other end

of the line is available and a connection with it can be established, the familiar login prompt will appear on your screen, and you will be able to log in to the other computer.

You can also call up the other system, specifying the speed of the connection. For this, enter

```
cu -lline -sspeed
```

Once again, *cu* looks in the *L-devices* file, this time to check that the requested speed for the requested line is available. If it is, the

```
Connected
```

message appears on your screen as before, and you can log in to the other computer.

Finally, you can also use *cu* without specifying either *line* or *speed*, but using *system-name* as found in the *L.sys* file. For instance,

```
cu doosy
```

calls the system *doosy* and goes through the login procedure as specified in the *L.sys* file.

Once you are logged in, you can transfer files, one by one, from your local machine to your remote machine by entering

```
~%put infile [outfile]
```

where *infile* is the name of the file you are transferring and *outfile* is the name you want it to have in the remote computer. If you do not specify *outfile*, the file transferred will have the same name as in the local computer.

You can also transfer files from the remote computer to your local one by entering

```
~%take infile [outfile]
```

This works exactly as *put* does, but in the reverse direction.

Although *cu* is the simplest method for file transfer between two computers, it has the disadvantage of not providing any error checking.

5. Automatic file transfer

The UUCP system also provides for an automatic file transfer capability between two computers. Other files, apart from the `L-devices` and `L.sys` files, have to be adjusted to permit this automatic transfer to operate properly. The remote computer must be set up to recognize your computer, login name, and password. Also to receive files from the remote computer, the local computer must be able to recognize the remote one as well.

5.1 Preparing the systems

5.1.1 The system node name

You must decide on a node name for your system. This is the name other people will put in their `L.sys` file to allow them to call your computer. If you want to change the node name from the current host name for your system, use a text editor to open the `/etc/HOSTNAME` file and change the first field in that file to the new name. When you reboot your system, the new node name will be in effect.

5.1.2 Dial-in and dialout ports

You need to modify the `/etc/inittab` file to enable gettys for dial-in ports and disable them for dialout ports. Depending on your exact needs, you might need a range of differing `/etc/inittab` lines. Here is a sampling that covers most normal cases:

- Outgoing calls only, at 9600 bits per second:

```
do:2:off:/etc/getty tty0 at_9600      # Port tty0
```

- Incoming calls only, at 1200 bits per second:

```
du:2:respawn:/etc/getty tty0 tt_1200  # Port tty0
```

- Calls in both directions, over an Apple Personal Modem.

```
do:2:wait:/etc/apm_getty tty0 # Set up port
```

```
du:2:respawn:/etc/getty tty0 mo_1200 # Port tty0
```

This last case deals with the fact that the Apple Personal Modem (APM) powers up with answering disabled. The `apm_getty` program forces the modem into answering mode.

5.1.3 Generic uucp logins

Your system comes configured with two generic uucp logins:

```
uucp::5:5:UUCP admin:/usr/spool/uucppublic:
        /usr/lib/uucp/uushell
nuucp::5:5:UUCP admin:/usr/lib/uucp:
```

Note that the first login has been wrapped onto two lines with the second line indented; this must appear as one long line in the `/etc/passwd` file.

Both of these uucp logins should be assigned passwords. Note that both generic logins have the same user and group IDs. See "The nuucp Login Environment," later in this appendix, for more information.

The nuucp login is for administrative use; when you are working on uucp you should log in as nuucp. Your home directory will be `/usr/lib/uucp`, and the permissions will be the same as on the uucp login. See "The nuucp Login Environment," later in this appendix, for information on using this login.

The other generic uucp login is uucp. The startup program for this login is `/usr/lib/uucp/uushell`, a script that establishes the time zone (TZ) variable and calls the `uucico` program. The startup program for any uucp login except the administrative login nuucp should be `uushell`.

5.1.4 System-specific uucp logins

It is not a good idea to allow other systems to log in as nuucp. Instead, you can setup a unique entry in `/etc/passwd` and `USERFILE` for each remote computer that will be calling your system. This allows you to control the access from each of these computers independently. For instance, if the password for one of these is inadvertently disclosed, you can change the password for that system's login and not have to inform the administrators of every computer with which your system connects. Likewise, if one computer calls at the wrong times or ties up the phone line, you can temporarily change the password to stop the problem until you contact the administrator of the problem system.

The conventional name for a system-specific uucp account is Uxxx, where xxx is the calling machine. For example, a typical set of entries in /etc/passwd might be:

```
uucp:SkQs1q/3e1NMo:5:5:UUCP:  
    /usr/spool/uucppublic:/usr/lib/uucp/uushell  
nuucp:GpBTy2Ls/upXk:5:5:UUCP admin:/usr/lib/uucp:  
Ufoo:avxoVAFxOzBTU:uid:5:UUCP for Foo:  
    /usr/spool/uucppublic:/usr/lib/uucp/uushell  
Ubar:4vpPMIdslZpHk:uid:5:UUCP for Bar:  
    /usr/spool/uucppublic:/usr/lib/uucp/uushell
```

Note: Three of the sample entries have been wrapped onto two lines with the second line indented. Each of these entries must appear on one long line in /etc/passwd.

Note that each system-specific uucp account should have a unique UID and the same GID as the generic uucp login.

If you have separate entries in /etc/passwd, you must have separate entries in USERFILE. When a remote system logs in, USERFILE is searched for the user name from /etc/passwd that was used to log in. The system name in the same entry must either match the system name of the remote computer or be null. See "Controlling Logins and File System Access," later in this appendix.

5.2 Receiving mail and files

You now want to make sure that users on the doosy system can send mail or files to users on your system. It is assumed that at least one modem port is a dial-in port, or that your one modem can serve as both dialout and dial-in modem. All you have to do is add an entry in the /etc/passwd file on your own system for the system doosy. The entry should look like this:

```
Udoosy::uid:5:UUCP for Doosy:/usr/spool/uucppublic:  
    /usr/lib/uucp/uushell
```

Note: This entry has been wrapped onto two lines with the second line indented. It must appear on one long line in `/etc/passwd`.

A machine that logs in as `Udoosy` will not get a normal shell but start up the `uucp` process directly with the program `uushell`, which in turn calls `/usr/lib/uucp/uucico` (UNIX-to-UNIX copy in copy out). Make sure that the user ID (*uid* in the example) is unique in the system and that the group ID is the same as the number in `/etc/group` in the entry for `uucp` (5 in the standard A/UX distribution). For more information on UIDs and GIDs, see Chapter 3, "User and Group Administration."

Next you must assign a password for the user `Udoosy`. While logged in as the root user, enter

```
passwd Udoosy
```

(You will be asked to enter the password twice, as usual for password changes.)

To give remote users access to parts of your file system, you must modify `/usr/lib/uucp/USERFILE`. A conservative security measure is to force all files to be copied in and out of `/usr/spool/uucppublic`. Note that the permissions on `/usr/spool/uucppublic` must be left open to all users (777), for example,

```
drwxrwxrwx 2 uucp uucp 944 Sep 24 08:49 uucppublic
```

Adding the following line to this file will allow all users on the system `doosy` to send you files on `/usr/spool/uucppublic` and to use the mail facility:

```
Udoosy,doosy /usr/spool/uucppublic
```

You must add a specific entry to `/usr/lib/uucp/USERFILE` for each system that is to have to have `uucp` access to `/usr/spool/uucppublic` on your system. Before your system can send mail and files to the remote computer, the same procedure must be followed on that system to allow your system access permission. See "Controlling Logins and File System Access," later

in this appendix for procedures that allow access to specific users or to other directories on your system.

5.3 Sending mail or files

To dial out, you must make sure that the modem port does not have a `getty` running on it, that is, that it is not working as an incoming modem. To do so, you must establish your modem either as an outgoing modem only or as both a dialout and a dial-in modem. See Chapter 7, "Managing Other Peripheral Devices."

Once the modem is able to dial out, you should be able to send mail to a user on the `doosy` system using the syntax

```
mail doosy!user
```

(When you are using the C shell, a backslash (\) must precede the exclamation point.)

When sending files using the C shell, you can use the notation `~uucp` as shorthand for `/usr/spool/uucppublic`. For example, to send a file named `my.file` to the `uucppublic` directory on a system named `doosy`, a user would enter the following commands from the C shell:

```
cp ./my.file /usr/spool/uucppublic
cd /usr/spool/uucppublic; chmod ugo+r my.file
uucp my.file doosy\!~uucp
```

See "Controlling Logins and File System Access," later in this appendix for procedures that allow access to specific users or to other directories on your system.

5.4 Cleaning up

If mail is sent from your computer to `doosy` and from `doosy` to your computer, you must make sure that `uucp` cleans up after itself; that is, that log files do not grow to enormous lengths, and that work spooled but not executed within a certain time (such as a few days or a week), is purged from the spooler. Three shell scripts are provided to do this work:

```
/usr/lib/uucp/uudemon.hr
/usr/lib/uucp/uudemon.day
/usr/lib/uucp/uudemon.wk
```

The `/etc/cron` utility runs these scripts on a regular basis; see `cron(1M)` in *A/UX System Administrator's Reference*. The `cron` utility starts when the system boots; see Chapter 9, "System Accounting Package." It checks the file `/usr/spool/cron/crontabs/uucp` (which you must create) once every minute to see if it has changed; see `crontab(1)` in *A/UX Command Reference*. Create `/usr/spool/cron/crontabs/uucp` as follows:

```
56 **** /bin/su uucp -c
    "/usr/lib/uucp/uudemon.hr > /dev/null"
0 4 *** /bin/su uucp -c
    "/usr/lib/uucp/uudemon.day > /dev/null"
30 5 ** 1 /bin/su uucp -c
    "/usr/lib/uucp/uudemon.wk > /dev/null"
```

Note: In this example, output lines have been wrapped onto two lines with the second line indented. These lines will each appear as one long line on the screen.

If you no longer want to run `uucp`, this file must be removed or renamed.

At this point, if all has gone well, you have a functioning remote mail system that cleans up after itself. Not only `mail` but also `uucp` and `uux` should work, because both these utilities use less of the `uucp` system than `mail` does. To confirm that everything is working, not just `uucp` and `uux`, establish a link with an actual computer. Next, send mail to a user on the other system, requesting that person to send you a reply. If you receive mail from the person on the remote system, everything is working properly.

If you do not receive a reply within a reasonable time, you should do a few things to find out where the problem lies. First test that the modem connections between the two sites are working correctly. For this, use `cu` to call up the other system, and ask the other person to call up your system with `cu`. Then, if the `cu` connection is working properly, check

whether the other person received your mail message and whether a response was actually sent back. Finally, you may have to go back through all the steps described in the subsections of "Automatic File Transfer" to check that everything was done properly, both at your end and at the other system's end.

6. Security and other tips

This section details the security features of uucp, some further administrative procedures, and the steps in debugging an improperly functioning UUCP link.

6.1 Controlling logins and file system access

If you have separate entries in `/etc/passwd` for each remote computer that will be calling your system (see "System-Specific uucp Logins," earlier), you must also have separate entries in `/usr/spool/uucp/USERFILE`. When a remote system logs in, `USERFILE` is searched for the user name from `/etc/passwd` that was used to log in. The system name in the same entry must either match the system name of the remote computer or be null. Null system names are not recommended.

An important security feature of `USERFILE` is its ability to restrict access to portions of the file system. For each line in `USERFILE` you can list directories for which remote systems will have access. Remote systems are then given access only to files in the listed directories. By default, remote systems have access to `/usr/spool/uucppublic`. The following example shows how you add for the remote system `doosy`, access to the directory `/usr/doosy`.

```
Udoosy,doosy /usr/spool/uucppublic,/usr/doosy
```

To give other users access to parts of your file system, you must modify `/usr/lib/uucp/USERFILE`. The following line in this file will allow all local users to send files and use the mail facility:

```
, /usr/spool/uucppublic
```

Notice the comma in the preceding line; this is a required part of the

syntax. The general format for entries in the USERFILE file is

[system], [login] directory

where *system* and *login* specify access permission to the named *system* and *login* names. When no system or user is mentioned, access is granted to all, but the comma that separates them in the general format must remain in place.

As an added security feature of USERFILE, you can use the *c* flag to force a call back to the remote computer instead of allowing it to log in to your computer. To use this feature, insert the letter *c* between the first and second entry on the line, for example,

```
Udoosy,doosy c /usr/spool/uucppublic,/usr/doosy
```

6.2 Conversation count checking

To improve security further, you can require a conversation count check every time a system calls. In other words, both systems must record the number of times they communicate with each other and check whether these counts coincide, as explained below. The file used for this purpose is */usr/lib/uucp/SQFILE*. Each line in this file contains the name of a system for which you will require the check. To initiate the checking, you need only enter the remote system name into the file as a separate line. The administrator of the remote system will have to add your computer's name to the *SQFILE* on that system. After the first call, the line might be

```
doosy 1 10/15-10:15
```

where *doosy* is the name of the other computer, *1* is the conversation count, *10/15* is the date, and *10:15* is the time of the conversation.

From that point on, the conversation count will be incremented on these corresponding lines every time these two computers are connected via uucp. If these counts do not match during an attempted call, the conversation will fail. Thus, to impersonate another computer you would need to know not only the login name and password but also the conversation count. If a call attempt ever fails because this file is corrupted on one of the two systems, all you have to do is reinitialize the lines (on both systems) so that they contain the system names only.

6.3 Controlling file forwarding

The uucp utility can forward files through intermediate nodes to get them to another system. If you plan to allow forwarding through your system (that is, make your system a node in the forwarding chain), and you want some control over this, you have to make entries in `/usr/lib/uucp/FWDFILE` and `/usr/lib/uucp/ORIGFILE`. The first file, `FWDFILE`, contains a list of systems to which you are willing to forward files via uucp. For instance, if you are willing to forward files to the computer `doosy` from other systems, just add a new line with the name `doosy` to the file. The second file, `ORIGFILE`, contains a list of systems and users from which you are willing to forward files. Thus, if you are willing to forward files originated in the `doosy` system by users `mark` and `marian`, you would add the following line to the file:

```
doosy,mark,marian
```

If these files do not exist, no restriction applies to forwarding on your system. This means that if you do not have a `FWDFILE`, you will allow forwarding to all systems to which you connect. Similarly, if you do not have an `ORIGFILE`, you will allow forwarding to originate on any system. To disable forwarding to any other system, create a `FWDFILE` with no contents (no forwarding permitted to any system). Similarly, you can create an `ORIGFILE` without any contents to prevent any other computers from using your system to forward files.

6.4 Controlling remote command execution

Another security feature of uucp is its capability to restrict the execution of remote commands. For uucp to execute remote commands, the names of these commands must be listed in the file `/usr/lib/uucp/L.cmds`, one command name per line. If you want maximum security, you should include a single line in this file with the command `rmail`, and no other line, so that no other remote commands can be executed. Without the `rmail` line, local users will not be able to get mail from remote systems.

6.5 File permissions

The last security feature is not part of uucp itself but involves the file permissions for the UUCP directories, executable files, and administrative files.

In general, the public should be denied read permission for most administrative files, especially `/usr/lib/uucp/L.sys`. This in turn requires that the owner of uucp's executable files be the same as the owner of the administrative files, which be given setuid permission in order to access them. See Chapter 3, "User and Group Administration." Thus, it is recommended that all these files be owned by uucp and that all the binary executables have the setuid bit set, as shown in the recommendations in this section.

The following modes are recommended for directories:

755	<code>/usr/lib/uucp</code>
755	<code>/usr/spool/uucp</code>
777	<code>/usr/spool/uucp/.XQTDIR</code>
777	<code>/usr/spool/uucppublic</code>
777	<code>/usr/spool/uucppublic/receive</code>

The following modes are recommended for binary files (notice the setuid permissions):

4111	<code>/bin/uucp</code>
4111	<code>/bin/uulog</code>
4111	<code>/bin/uuname</code>
4111	<code>/bin/uustat</code>
4111	<code>/bin/uusub</code>
4111	<code>/bin/uux</code>
4111	<code>/usr/lib/uucp/uucico</code>
4111	<code>/usr/lib/uucp/uuclean</code>
4111	<code>/usr/lib/uucp/uuxqt</code>

The following modes are recommended for script files:

755	<code>/usr/bin/uupick</code>
755	<code>/usr/bin/uuto</code>
400	<code>/usr/lib/uucp/uudemon.day</code>
400	<code>/usr/lib/uucp/uudemon.hr</code>
400	<code>/usr/lib/uucp/uudemon.wk</code>
755	<code>/usr/lib/uucp/uushell</code>

The following modes are recommended for uucp system files:

444	/usr/lib/uucp/ADMIN
444	/usr/lib/uucp/FWDFILE
444	/usr/lib/uucp/L-devices
444	/usr/lib/uucp/L-dialcodes
444	/usr/lib/uucp/L.cmds
400	/usr/lib/uucp/L.sys
444	/usr/lib/uucp/ORIGFILE
400	/usr/lib/uucp/SQFILE
400	/usr/lib/uucp/USERFILE

6.6 The nuucp login environment

A user login for nuucp is set up in the /etc/passwd file in the A/UX standard distribution. You should always log in as nuucp to do UUCP administrative work, because working as the root user can be dangerous and is unnecessary when you deal with UUCP system files.

The administrative login entry in /etc/passwd is set up as follows:

```
nuucp::5:5:UUCP admin:/usr/lib/uucp:
```

Note that the directory /usr/lib/uucp has been chosen as the home directory and that the default /bin/sh is the startup shell. If the startup shell were /bin/csh or /bin/ksh, this would appear as the last field on the line. Note also that the nuucp login has the same user and group ID as the uucp login. Because the uucp login occurs first in this file, all files created by the nuucp administrative login will be owned by uucp. This is recommended.

You should assign a password to the nuucp login, and you may also create a .profile file in nuucp's home directory with some helpful shell procedures as follows:

```
cdlib  () { cd /usr/lib/uucp; }
cdpub  () { cd /usr/spool/uucppublic; }
cdspl  () { cd /usr/spool/uucp; }
poll   () { /usr/lib/uucp/uucico -rl -s$1 &; }
pollx  () { /usr/lib/uucp/uucico -rl -s$1 -x4 &; }
rmstat  () { rm -f /usr/spool/uucp/ST*; }
taillog () { tail -f /usr/spool/uucp/LOGFILE; }
```

As you get to know UUCP, you will find out how helpful these procedures can be.

The file `/usr/lib/uucp/ADMIN` consists of a list of systems and their descriptions separated by a tab. The entry for the system `doosy` might look like this:

```
doosy      1234 University Avenue,  
Sometown, CA
```

Now when you run `uuname` with the `-v` option, the description will also be displayed for `doosy`.

Three UUCP commands are for administrative as well as general use:

```
/bin/uulog  
/bin/uustat  
/bin/uusub
```

You can use the `uulog` command to display selected portions of `/usr/spool/uucp/LOGFILE`. You may request lines for either a specific system name or a user. For example, to check what has happened with the system `doosy`, issue the command

```
uulog -sdoosy
```

You can use the `uustat` command to find the job number of a UUCP request and also to cancel a UUCP request. It is recommended that you use this method to terminate UUCP jobs if at all possible. For example, if, during a transfer, you discover that you have requested the wrong file, you first run the following command to get the job number:

```
uustat -sdoosy
```

The status of requests is displayed in reverse chronological order, and so your request is near the top. If the job number is 1234, you run the following command to cancel the request:

```
uustat -k1234
```

You can use the `uusub` command to collect and display statistics for the systems with which your system communicates.

Before statistics can be collected for a system, uusub must first recognize it by adding it to its list. The command to do this is

```
uusub -adoosy
```

You can also use uusub to connect to a system. This is usually scheduled by cron. If you want to call the doosy system every hour on the half hour, the `/usr/spool/cron/crontabs/uucp` entry should look like this:

```
30 **** /bin/su uucp -c "/bin/uusub -cdoosy > /dev/null"
```

See Chapter 9, "System Accounting Package," for a thorough discussion of the format of cron entries.

6.7 Suggested links

The last issue concerning administration is purely conventional. Instead of using names such as `tty0` in the files `L.sys` and `L-devices`, it is easier to make links in the `/dev` directory to provide alternate names for ports. The naming convention is used for ports that either have direct links to other computers or have modems attached. For instance, if `tty0` is connected to a modem and `tty5` is connected directly to doosy, make the following links:

```
ln /dev/tty0 /dev/acu.hayes
ln /dev/tty5 /dev/dir.doosy
```

Then use `acu.hayes` instead of `tty0` in the administrative files and `dir.doosy` instead of `tty5`. This also makes it easier to use `cu` because you don't have to remember the number of the port. See Chapter 5, "Using cu" in *A/UX Communications User's Guide*. If the connection is changed to another port, all you have to do is remove the link to the old port and link the name to the new port. None of the uucp administrative files need be modified.

6.8 Troubleshooting uucp

There are several things you can do if uucp is not working properly on your system.

If you have no information, such as status files or log files, and you have to figure out why uucp is not working and fix it, the first thing to do is to make sure the port, modem, and phone line are working. You can use the `cu` utility to determine this. See Chapter 5, "Using cu" in

A/UX Communications User's Guide. The command to check `tty0` would be as follows:

```
cu -l tty0 dir
```

If you cannot even get the Connected message from `cu`, the port probably has the wrong permissions. To correct this, enter the following command:

```
chmod 666 /dev/tty0
```

At this point you should be able to communicate with the modem and have it dial up the other system. When you get the login and password prompts, use the login and password in `L.sys`. The remote system should respond with something like this:

```
Shere=doosy
```

This means you have reached the `uucico` shell on the remote system and you can do no more using `cu`. Otherwise, contact the administrator of the remote system and explain that your `uucp` login is not working.

Once you have established that the port, modem, and phone line are working, test the connection with the following command:

```
/usr/lib/uucp/uucico -rl -sdoosy -x4 &
```

Note: You should always run this command in the background so you can kill it if it hangs. Do not use the `-9` option of `kill` because `uucp` will not catch the signal and clean up after itself. Instead, use `kill` with no options.

To save the debug output in a file, use the `script` program:

```
script /usr/lib/uucp/uucico -rl -sdoosy -x4 &
```

By comparing the debugging output of this command with the *expect-send* sequence in the `L.sys` file, you can usually tell if something is wrong. For instance, if an entry in the `L.sys` file specifies that the password is `foo` but the password that appears in the debugging output is `fee`, you can then modify `L.sys` and keep testing until it works. By running the `uucico` command with the `-x9` options, you may

generate a lot more debugging output.

A typical problem is finding strange times in the log files. This generally happens because the time zone environment variable is not set correctly. You can avoid this by making sure that `/usr/lib/uucp/uushell` is the startup script for all uucp logins. The contents of this script are

```
exec env TZ=PST8PDT /usr/lib/uucp/uucico
```

Make sure that the setting for TZ corresponds to your own time zone.

Most problems with uucp occur because of incorrect permissions on files. You should always check this if uucp starts working improperly. See "File Permissions," for specific recommendations about the appropriate permissions for the files used by the UUCP system, and Chapter 3, "User Administration," for a general overview of file permissions.

PLACE TAB HERE

NOTES

Back of Tab

Place Tab Here

INDEX

Back of Tab